Stochastic Learning of Textures

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften

vorgelegt von
Ing. Comp. Oscar Antonio Zárate Aguila
aus Ocotlán, México

genehmigt von der
Mathematisch-Naturwissenschaftlichen Fakultät
der Technischen Universität Clausthal

Tag der mündlichen Prüfung

17.12.2004

Die Arbeit wurde angefertigt am Institut für Informatik
der Technischen Universität Clausthal

Dekan:	Prof. Dr. D. Mayer
Berichterstatter:	Prof. Dr. I. Kupka
Mitberichterstatter:	Prof. Dr. K. Ecker

Zusammenfassung

Oscar Antonio Zárate Aguila

Stochastisches Lernen von Texturen

Bei der Rekonstruktion fehlerhafter Bilder tritt das Problem auf, Bildlücken mit einem passenden Bildinhalt zu versehen. Oft handelt es sich dabei um Hintergrundinformation, die im Prinzip aus der Umgebung rekonstruierbar ist. Interpolationsverfahren hinterlassen oft schwer zu glättende Ränder der Fehlstelle. Da viele Hintergründe Zufallsmuster enthalten, liegt es nahe, nach einer Möglichkeit zu suchen, diese stochastisch zu erfassen und die gewonnene Information zur Generierung wieder einzusetzen. Ein solches Verfahren könnte in der Lage sein, Bildinformationen stochastisch wahrzunehmen und ähnliche Bildinformationen zu erzeugen. Dies hätte Auswirkungen sowohl auf die Speicherung von Bildinformationen als auch für die Erzeugung künstlicher Bilder.

Zur Charakterisierung von Texturen werden bereits seit einiger Zeit Statistiken in Form der sogenannte Cooccurrence-Matrizen verwendet. Diese dienen nicht der Wiederherstellung von Bildinformation sondern der vergleichenden Charakterisierung. Durch Kombination der Informationen aus mehreren solcher Matrizen lässt sich ein Bildaufbauprozess für die Textur konstruieren. Im Rahmen des Dissertationsvorhabens soll nun untersucht werden, welche Qualität sich hierbei erreichen lässt und ob das Verfahren mit Methoden der Vektorquantisierung konkurrieren kann.

In der Arbeit wurden zwei Verfahren untersucht: erstens, die Cooccurrence-Matrizen (CM) und zweitens, die Vektorquantisierung (VQ). Bei der ersten Methode wird eine Nachbarschaft von Cooccurrence Relationen festgelegt und die CM berechnet, um die Information des Bildes zu sammeln. Danach werden die Texturen mittels der Matrizen erzeugt. Eine Bewertung der Reproduktionsqualität ist möglich durch die Entropiewerte, die aus der CM berechnet sind.

Bei der zweiten Methode werden, als erstes, der Vektorraum und die Anzahl Codevektoren festgestellt. Zunächst wird ein Stochastischer Durchlauf aller Lernvektoren ausgeführt, um die Grundlage der Rekonstruktionsverfahren zu schaffen. Die Erzeugung der Muster wird durch die Anwendung der Codevektoren durchgeführt.

Beide Methoden wurden mit verschiedenen Mustern getestet und miteinander verglichen. Andersartige Erzeugungsverfahren wurden auch analysiert und verglichen.

Nach der experimentellen Auswertung wurden folgende Ergebnisse erzielt: Unabhängige Produktion von Texturen und die Erkennung einen Teilbildrelation durch Occurrence-Matrizen.

Abstract

Oscar Antonio Zárate Aguila

Stochastic Learning of Textures

By the reconstruction of defective images, appears often the problem of filling the irregularities with an appropriate pattern. Normally, the information is part of the background, which can be restored from the neighbourhood. Interpolation procedures leave edges of the defect, which can be smoothed difficulty. Since many backgrounds contain random samples, it leads to the search of a possibility to acquire this information stochastically and use it for the reconstruction. Such a procedure could be able to notice picture information randomly and produce similar picture information. This would have effects on the storage of picture information and the production of artificial pictures.

For the characterisation of textures, statistics are used already for some time in form so-called Co-occurrence Matrices. These do not re-establish the picture information, but separate it and characterise it. By combining such matrices, a screen layout process for the texture can be designed. In this thesis, it is examined which quality can be reached with this method and whether the procedure can compete with methods of vector quantization.

In this work two procedures were examined: first of all, the Co-occurrence Matrices and secondly, Vector Quantization. With the first method a neighbourhood of Co-occurrence relations is specified and the matrix is calculated, in order to collect the information of the picture. Afterwards the textures are produced from the obtained matrices. An evaluation of the reproduction quality is possible by the entropy values, which are computed from the co-occurrence matrices.

With the second method, the vector space and the number code vectors are determined. First a stochastic run of all learning vectors is implemented, in order to create the basis of the reconstruction procedures. The production of the samples is accomplished by the application of the code vectors.

Both methods were tested with different samples and compared with each other. Different production methods were also analysed and compared.

After the experimental evaluation the following results were obtained: Independent production of textures and the recognition of field relations by Co-occurrence Matrices.

A mis padres,
mis hermanos
y mi familia

"Resígnate a no haber podido hacer una cosa, más nunca a no haberlo intentado, si vale la pena de intentarlo."

Amado Nervo

"Si en la lid el destino te derriba; si todo en tu camino es cuesta arriba; si tu sonrisa es ansia insatisfecha; si hay faena excesiva y vil cosecha; si a tu caudal se contraponen diques; date una tregua pero... ¡no claudiques!"

Rudyard Kipling

Table of Contents

1 Introduction

Nowadays Image Processing is one of the most important branches of Computer Science. Going deeper into this field, one of the most important areas is the learning and reconstruction of textures or patterns. The fields covered by this theme are almost incommensurables, among the most known for its dependence on textures recognition are, Chemistry, Medicine and Criminology. There are several methods that analyse and process different patterns for the desired necessities of each field.

The goal of this work was the search of different learning alternatives of textures guided by texture reconstruction problems and restoration. It is not intended to develop new operators for such manipulations, but looking for general methods of learning textures which could be the basis for treating textures in an intelligent system.

In this thesis, two stochastic methods are studied for the learning, and restoration, of textures. The first approach uses Co-occurrence Matrices for the obtaining of information and restoration of images. The second approach takes Vector Quantization as the main tool of the learning process and later for the reconstruction of the desired texture.

Until now Co-occurrence Matrices are used mainly as an image edition tool, whose features are edge detection, sharpening of the image, etc. Co-occurrence matrices are also used for characterisation of textures, but not for image production or generation, which is one of the features included in this thesis. The most common Co-occurrence Matrix (Grey Level Co-occurrence Matrix) handles only monochromatic textures; in this work this method is enhanced for the processing of chromatic textures.

Vector Quantization is a neural network type whose intended use is classification. However, it is also used as lossy compression method for image and signal processing. In this work, another feature is implemented for Vector Quantization, image generation.

In chapter two of this work, a brief introduction to human and computer vision is described. This gives the basis to the creation of colour models, specifically RGB model, which is the role model used in this thesis for the image processing. Co-occurrence Matrices and Vector Quantization methods use RGB model for extraction of image information as well as for image generation and reconstruction.

Chapter three gives a more detailed description of Co-occurrence Matrices, their origin and main features, as well as some examples to clarify the *modus operandi* of this method.

The experimental part of this thesis is located in chapter four. The first section of this chapter shows the Co-occurrence Matrices analysis, its modus operandi and obtained results. The second section shows the Vector Quantization process, as well as the results obtained with this method.

In the Co-occurrence Matrices method, the image information is captured using co-occurrence matrices. Three different types of co-occurrence matrices are obtained, one for each colour channel (red, green and blue), which allow the learning and restoration of not only black and white images, but also chromatic textures. Once the matrices learn the texture properties, distribution curves are calculated for the later restoration of the image.

The restoration process selects randomly the most suited colour from these distribution curves. Important parameters for the correct reconstruction of textures are the working area, which scans the image and obtains the co-occurrence matrices, and the entropy filters, which select the important colour information to be used in the distribution functions. Once this process finishes, a regeneration of the original texture is obtained from the previously learned information.

The second method obtains the information with the help of vector quantization. An initial number of vectors are selected randomly and acquire the image information. These vectors are trained and refined, according to the established methods of Neural Networks. At last, colour distribution curves (one for each channel) are calculated from the initial code vectors. The image is restored using the distribution functions to select the most appropriate colour shade. The parameters needed for the accurate learning and restoration of textures are the number of initial vectors and their working area, which obtain the image information; the second parameter is the number of epochs, which define the vector training; the last parameter is the learning factor, which is the distance increment that the winner vectors covers when it approaches to the comparative vector in the training phase. The result of this procedure is a reconstruction of the original analysed sample based in the previously acquired information.

Both methods are analysed and compared in chapter four, in order to determine the most suitable option for the learning of textures.

The fifth chapter shows the conclusions and further work to be done with both methods. The list of algorithms is located in chapter six and the list of references in chapter seven.

2 The Vision

2.1 Human Vision

The human is a visual being. Among the animals, the human is one of the beings who depend at most on his eyes. Human beings achieve more than 99% of the environment information due to the sight. Dogs have extraordinary sense of smell and hearing, bats use high frequency sounds, snakes locate their pray thermo graphically and fish have organs, that perceive, and sometimes also generate electrical fields. Birds are also visual creatures, but they do not have the same configuration of human visual system. Their eyes are located on opposite sides of the head, which provides 360-degree coverage, but almost nothing in stereopsis, also known as three-dimensional vision. Birds have four or five colour receptors, while the humans only have three, green, red and blue.

People with sight problems wear glasses due to the dependence of that sense. However, hearing deficiency is highly tolerated before buying an appropriate equipment to correct the problem; speaking of other senses, there are not even prosthesis or devices to substitute them.

The human eye consists of many parts responsible to process reflected light and generate the image, which is interpreted by the brain. The eyelids and eyelashes protect the eyes from tiny dust particles and at the same time moist the cornea, avoiding the opacity of it. The cornea is the outside part of the eye. The light enters through the cornea, goes trough aqueous humor and arrives to the pupil. The iris is a pigmented muscle, which contracts to diminish the pupil or relax to enlarge it.

The environment light intensity determines the size of the pupil. In a lit place, the pupil diameter is very small, avoiding the entrance of a big amount of light, which could cause damages to the eye. Otherwise, in a dark room, the pupil size grows to let through a greater quantity of light to the eye, allowing to see the objects that surround us. The next figure shows the anatomy of the human eye.

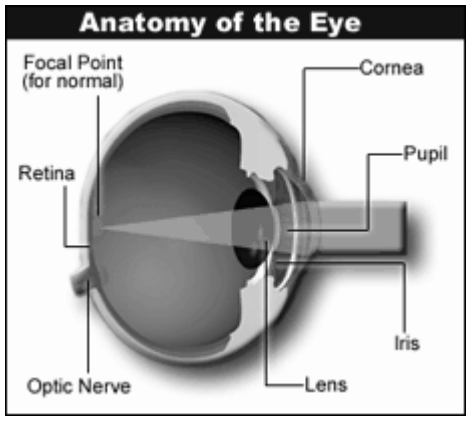


Figure 2.1.1 Anatomy of the human eye.

Once the pupil has filtered the light that comes from outside, the crystalline lens projects the acquired image at the back of the eye, *videlicet* the retina. The crystalline lens changes its form permitting the eye to focus at different distances.

In the course of time, this adjusting process becomes difficult, therefore, (it) is necessary the use of glasses. The projected image in the retina is inverted; the brain interprets it and converts it to the original form. For more information about human vision, see [1, 2, 3].

The most common eye diseases are shortsightedness, long-sightedness and astigmatism: Shortsightedness occurs when the light rays converge in front of the retina. This can be caused for several reasons; for example, the eyeball is too long or the lens does not work properly. Divergent lenses are needed to correct this problem; they spread the light, causing the image to be projected at the retina and not in front of it.

Long-sightedness is precisely the opposite; the light rays are focused behind the retina. To correct this disease, convergent lenses are required; they change the course of the rays, causing them to converge at the retina.

Astigmatism is generated by a cornea malformation. The astigmatic person sees the images or objects twisted. Cylindrical lens are used for this defect; they deviate certain light rays more than others, counteracting the cornea irregularities.

In order to process the image in the brain, the retina has cells called photoreceptors, which are stimulated in different ways. There are two types of them: cones and rods, named that way because of their shapes, see figure 2.1.2. Cones are photosensitive elements responsible for the chromatic sensations (colours), visual sharpness and also sensitive to bright stimulus. Cones have photo pigments, which are particularly responsive to green, red and blue colours. According to Murch[4], 64% of the cones have red photoreceptors, 32% contain green photo pigments and only 2% have blue photoreceptors.

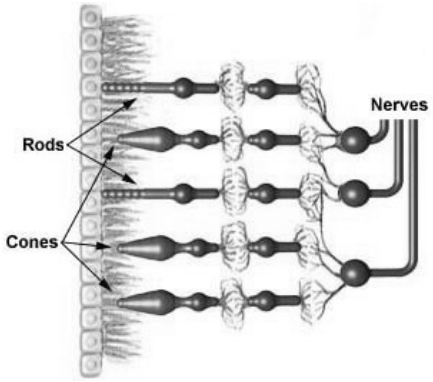


Figure 2.1.2 Rods and cones.

When the light is reflected on a high concentration of cones, these have a spatial resolution of the image, which permits the detection of even the slightest colour variation. Rods, as well as the cones, are photosensitive cells but only to the presence or absence of light, that is, achromatic sensations (white, black and gray) and luminous stimulus.

Human eye has approximately 8 millions cones. The majority of them are located in the central part of retina, called fovea. The number of rods is much higher, around 120 millions, and they are mainly distributed in the retina periphery. Figure 2.1.3 shows the arrangement of rods and cones across the retina.

Due to this distribution, human eye is more susceptible to light than colour; therefore, to obtain image details, it is necessary to focus directly to them. Following a simple experiment: by laying one hand in the opposite page of this page and focusing in this sentence, it is not possible to recognize hand details like skin texture, hair or the tiny wrinkles of the joints. Only when the hand is directly observed, putting this way the image in the middle of the retina, the detail level will be increased. More information about human vision can be found in [5, 6].

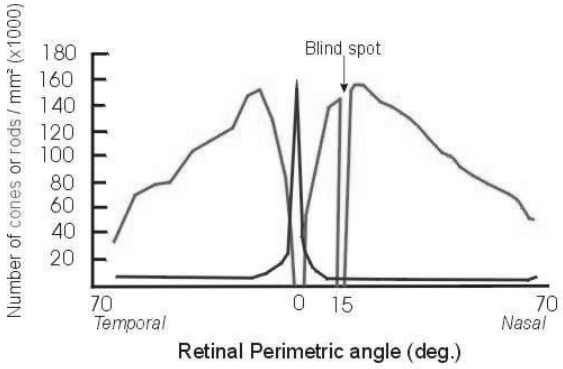


Figure 2.1.3 Arrangement of cones and rods across the retina.

2.2 Computer Vision

Computer vision is based on the human visual system. The main difference between them lies on how the images are acquired. Cameras are the most common devices for image acquiring. The main categories are photographic and video cameras, but they have a large list of subcategories, depending on the resolution, the materials used for the image capture, the method to acquire the objects (especially in video cameras). There are other devices like scanners or digital drawing palettes. They have all one thing in common. The image they obtain must be transformed to an electrical signal, in order to be analysed or processed by the computer.

The first computer vision systems were created over 30 years ago and they had great difficulties mainly because of their limited computing power. The classic architectures of von Newman-type computers, known for their sequential processing, have stopped the development of computer vision systems, where a parallelism for the image processing is required.

There are three hardware requirements for the computer vision system, first, a device for obtaining images, second, a digitizer that converts the picture to electric signals and the last requirement is a processing system, which permits the displaying of the acquired signal.

The image processing systems generally have two levels. There is a first pre-processing stage, where low-level operations are done, like feature extraction, filtering, etc. The second stage is in charge of the analysis and interpretation of the pictures, depending on the desired application. More information can be found in [7].

Digitalization is the process that converts the image information from analog to digital; that is, the image in the way humans see it, to signals that can be interpreted and processed by computers.

2.2.1 Image Visualization

The most common image displays are the video monitors. There are various types of them; the most common are Cathode Ray Tubes (CRT) and Liquid Crystal Displays (LCD). Both models are described in the following sections.

2.2.1.1 CRT

The CRT is a vacuum tube able to generate images. These result when an electron beam (cathode rays) strikes the phosphor-coated screen. The phosphor, when excited, emits a small light dot in every position where the electron-beam strikes. The majority of computer monitors are CRTs.

At the back part of the picture tube is located an electron gun, whose main parts are the cathode and the control grid. By supplying current to the heating filament inside the cathode, the temperature increases, provoking the electrons to be separated from the cathode surface. In the tube vacuum the free electrons are accelerated to the phosphor coating by a high positive voltage. This voltage can be obtained with an acceleration anode or with a metallic coating at high positive voltage. Sometimes, in the electron gun, both the focusing system and the acceleration anode are mounted.

Modifying the control grid voltage can regulate the beam intensity. By applying a high negative voltage, the beam is cut, because the electrons are repelled. A less high negative voltage on the grid reduces the electron number that goes to the screen. The quantity of emitted light of the phosphor-coated screen depends on the number of electrons that strikes against it.

The focus system forces the electron beam to converge into a small area, at the moment it strikes the screen. Without focusing, the electrons could repel each other and the shaft would expand when approximating to the screen. This focus system is made with electric or magnetic fields. A metal cylinder with positive charge forms an electrostatic lens, that focuses the electron beam to the center of the display.

The distance that electrons cover varies according to their position on the monitor. Due to the monitor curvature, the distance is bigger if the electrons go away from the center of the screen, which gives a bad resolution, displaying the image blurred at the extremes of the monitor. Only the image at the center of the screen is optimum, because this is the ideal focus.

There is also a deflection system that is made of coils mounted at the exterior of the CRT. Two sets of coils are used, a pair of them is placed vertically and the other pair horizontally. The magnetic field generated by the coils produces a deflection force, which is transversal and perpendicular to both, magnetic field direction and electron shaft direction. Deflection levels are adjusted by controlling the circulating current in the coils.

When electrostatic deflection is used, two pairs of parallel sheets are mounted in the CRT. A pair of sheets, placed horizontally, controls the vertical deflection and the other pair, placed vertically, is in charge of the horizontal deflection. Figure 2.1.4 shows the components of a CRT.

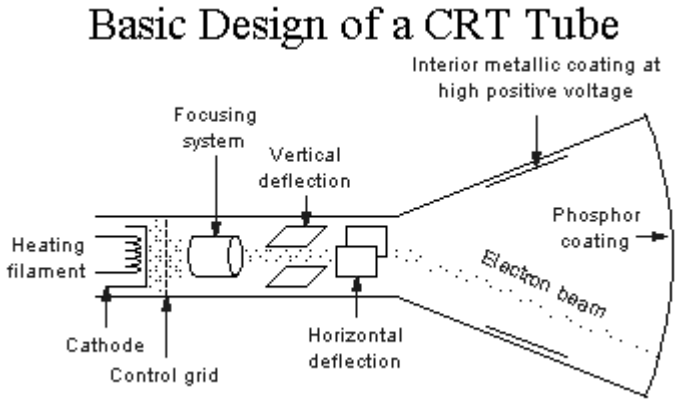


Figure 2.2.1 Cathode ray tube.

Light dots are produced on the monitor due the energy transfer from the CRT to the phosphor coating. When the electrons strike the coating, the phosphor stops them and absorbs their kinetic energy. By friction, part of the shaft energy is converted in thermal energy, the rest of it provokes the electrons in phosphor atoms to move to higher energy quantum levels. After that, the excited phosphor electrons return to their stable state and emit the additional energy as small quantum of luminous energy. The image on the screen is the combined effect of all electron light emissions. The light frequency emitted by the phosphor is proportional to the difference of energy between the state of steady quantum and the excited state.

There are various types of phosphor, their main difference, besides colour, is the persistence, *videlicet*, how much time will they continue emitting light, after the electron beam is removed. Persistence is defined as the time the emitted light takes to diminish to one-tenth part of its original intensity. Phosphors with low persistence are used mainly in animations; phosphors with high persistence are useful to display complicated static images full of details.

Although some phosphors have persistence higher to one second, graphic monitors are built with phosphors, whose persistence lies in the order of 10 to 60 μs. For more information about CRTs see [8, 9]. There are different types of picture tubes, the raster scan displays, random display devices, polychromatic CRTs, etc.

2.2.1.1.1 Raster Scan Display Devices

This is the most used type of display in the picture tube family. Its operation is based on television technology. In this system, the electron shaft covers one scan-line at the time from top to bottom.

The generated image is saved on the frame buffer, or refresh buffer. This memory area contains the intensity values for each screen dot, to wit, pixels. The information storage capacity makes the raster scan displays the best option to project realistic scenes with faint shadowing and colour patterns. Televisions and printers use also this technique.

The intensity range of pixels position depends on the working system. In a black and white system, each screen dot has two states, on and off; only one bit is necessary to control the intensity of the screen positions. For a two level system, the bit value 1 activates the electron beam and bit value 0 deactivates it.

To display variations in colour and intensity, additional bits are needed. In high quality system up to 24 bits per pixel are included, which represents a great amount of storage for the image buffer, depending on the system resolution. Systems of 24 bits per pixel and resolution of 1024x1024 require 3 megabytes of space for the image buffer, also known as bitmap.

The image refreshing in these systems is done every 60 or 80 frames per second, although some systems use higher refresh rates. Generally, the refresh rate is described in units of cycles per second (Hertz), where every cycle represents one frame. At the end of every scan-line, the beam returns to the left side of the screen. This return is called horizontal retrace. At the end of every frame (displayed at 1/80 to 1/60 second) the shaft returns to the upper left corner to start with the next frame. This is known as vertical retrace. Figure 2.2.2 shows how a raster display works.

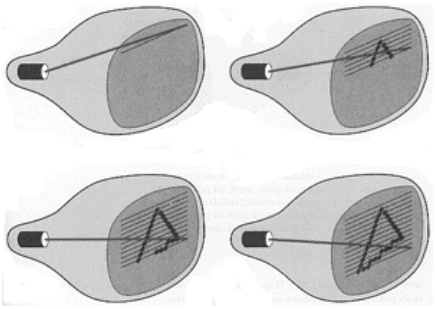


Figure 2.2.2 Raster scan display process.

In some raster systems, televisions also, each frame is displayed in two steps by a method known as interlacing. In the first step, the shaft covers the odd lines from top to bottom. After the vertical retrace, the shaft covers the even lines. By using this method, the whole screen is displayed in half the time required to cover all the lines in one step. Interlacing is used mainly when there is a low refresh rate, for example 30Hz, where some flickering is present. By using interlacing, every step is displayed in 1/60 second, which comes near the refresh rate of 60Hz. The effectiveness of this technique lies in the information similarity between adjacent lines, thanks to this factor, the image can be recognized. For more information see [8].

2.2.1.1.2 Random Scan Display Devices

By this type of monitors, the CRT focuses the electron beam only to the screen parts where the image is projected. These systems generate the image tracing one line at the time, which is the reason they are called vector display devices. The lines that form the image are refreshed by means of random tracing in any specific order.

The retrace index depends on the number of lines to be projected. The image is saved as a set of commands for line tracing; these instructions constitute the display refresh file. To show the image, the system pass through a cycle of the commands set and traces the lines one by one. After the process, the system repeats the cycle until the first line. Figure 2.3.3 illustrates this process.

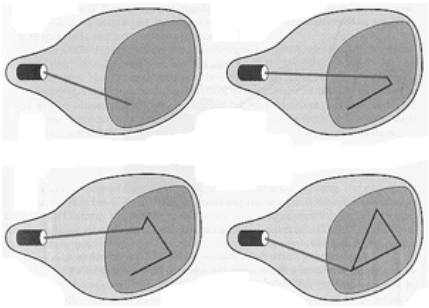


Figure 2.2.3 Random scan display process.

These displays are designed to trace all image lines in 30 to 60 times per second. High quality vector systems are able to process around 100,000 short lines at that refresh rate. When a small image is projected, the system delays in every refresh cycle to avoid indexes higher than 60Hz, otherwise, the frequent refreshing of lines can burn the phosphor coating.

Random scan displays cannot project shadowed realistic images. However, vector displays have generally better resolution than the raster display systems, because they save the image as a set of instructions and not as intensity values. These displays produce smoother line tracing because the electron shaft follows the line trajectory. In raster systems, the lines are not straight because they are traced as a set of discrete dots. For more information about these displays see [8].

2.2.1.1.3 Colour Monitors

A CRT monitor is able to display colour images by using a phosphor combination that emits polychromatic light. By combining the emitted light of different phosphors, a wide range of colours is obtained. There are two basic techniques to display colours in CRTs, the first one is the shaft penetration method and the other one is the shadow mask.

The shaft penetration method is used in random scan displays. The interior of CRTs is covered with two phosphor coats, generally red and green. The resulting colour depends on how much the electron beam penetrates the phosphor coating. A slow shaft excites only the outer red coat, while a fast beam goes through the red coat and excites the inner green coat. By intermediate shaft velocity, combinations of green and red are emitted, to wit, yellow and orange. The beam acceleration voltage is in charge of controlling velocity and screen colour. This technique shows an economic way to build random scan colour monitors, but only four colours can be created and the resolution is not optimal.

Shadow mask method is used in raster scan systems (including colour televisions) and produces a much higher variety of colours than the shaft penetration method. Shadow-mask CRTs have three coloured dots for each pixel. These phosphors emit red, green and blue light. This type of monitor has three electron beams, one for each colour dot, and a shadow mask, just behind the phosphor coated screen.

Another very often used method in colour monitors is the delta-delta. In this method, the electron beams are focused to the shadow mask, which has a succession of holes that are lined up with the phosphor dots. When the shafts go through one hole, they activate a triangle of dots that appears as a small colour spot on the screen. Phosphors are ordered in such way, that each beam can only activate its corresponding colour when going through the shadow mask. Figure 2.2.4 shows the operation of a colour CRT with the delta-delta method.

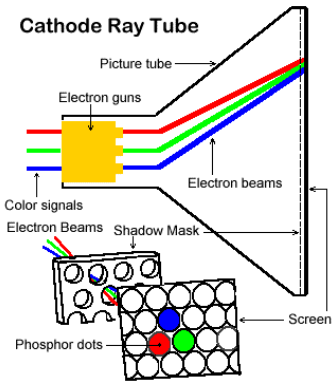


Figure 2.2.4 Colour CRT.

Another way to do it is with an inline order, where the electron beams and the red, green and blue dots are arranged along a scan line, instead of a triangular pattern. This beams arrangement is easier to maintain aligned and is used regularly in high resolution CRT.

The intensity level variations of electron shafts cause the colour changes on the monitor. The combination of shaft intensity produces a light spot for each pixel position, since the eyes tend to mix the three colours into one dot. The perceived colour depends on the excitation grade of red, green and blue phosphors. A white (or gray) area is the result of activating the three dots with the same intensity. Magenta is obtained by the red and blue dots, cyan with the combination of green and blue and by mixing red and green, yellow is obtained.

In low cost systems, the electron beam can be only on or off, limiting the display to eight colours. More complex systems can handle intermediate intensity levels for the electron beams, allowing the creations of millions of different colours.

Colour CRTs are designed as RGB monitors. These use the shadow mask method and take the intensity level for each electron shaft (red, green and blue) directly from the computation system, without any intermediate process. Graphic systems for high resolution scanning have 24 bits per pixel in the image buffer and can handle 256 voltage levels for each beam, so, they are able to produce up to 17 millions of colours for each pixel.

A RGB system with 24 bits per pixel is known as a total colour system or real colour system. For more information about colour CRTs see [8, 9].

2.2.1.2 Flat Panel Displays

These devices require less volume, weight and energy than CRT monitors. The main characteristic about flat panels is precisely that they are much thinner than CRTs, which opens a wide range of new possibilities, for example, to hang the panel on the wall, use it on the wrist or write directly on them like pads. Laptop computers and pocket video games are some other applications of the flat panels.

These displays are separated in two categories: light-emitting and non-light-emitting panels. The light-emitting displays transform the electric energy into light. Some examples of these are the plasma panels and thin-film electro-luminescent devices. The non light-emitting displays use optic effects to convert the received light into graphic patterns. The most representative examples of these systems are the liquid crystal displays (LCD).

Plasma panels are also known as charged gas displays; the main components are two crystal plates filled with a special gas mixture, which usually includes Neon. A set of conduction stripes is mounted in one of the plates in vertical position. Another set is placed horizontally in the second plate. The voltage applied to both vertical and horizontal conductors causes the gas to be decomposed in shining plasma of electrons and ions. The image is saved in a tracing buffer and the voltages refresh the pixels positions 60 times per second. Alternating current is used to offer a faster application of turning voltages on, resulting in brighter displays. The electric field of the conductors causes the pixel separation. More information about plasma displays is located in [8, 9, 10].

An additional advantage of plasma panels over CRT monitors is that thin displays do not have the slight curvature of the cathode ray tubes, which results in a distortion-free image displaying on the extremes of the screen. The main disadvantage of this technology is the price; they are much more expensive than normal chromatic CRTs. Next figure shows an example of plasma panel.

Figure 2.2.5 Image displaying in a plasma panel.

The thin-film electro-luminescent devices work similar to plasma panels. The main difference is that two plates of glass are filled with Zinc sulfur combined with Manganese instead of a gas. By applying voltage to a pair of transversal electrodes, the compound is converted in a conductor. After that, Manganese atoms absorb the electric energy and liberate it as photons, which cause a similar effect to the shining plasma. These displays require more energy than the plasma displays and the resolution is not so good as in panels or CRT.

Liquid crystal displays are used in small systems like calculators and laptops. These non-light-emitting devices produce images when polarized light of the environment passes through a liquid crystal material that is able to transmit light. This technology, as well as plasma panels, allows the creation of thin monitors, although they work in a different way. To generate an image, the LCD blocks the received light instead of emitting it. For this reason, LCDs need less energy than plasma displays.

Inside a LCD, two crystal plates press the liquid crystal material, each of them with a transversal light polarized panel. Horizontal conductors are mounted in one plate, and vertical conductors in the other. The intersection of conductors defines the pixel position. By turning on the monitor, the molecules are aligned according to the polarized panels, provoking the light to “twist” from one panel to the other; the resulting light is the one the observer perceives. To deactivate a pixel, voltage to the intersected panels is applied, aligning the molecules so that the light does not twist.

This type of display is known as passive matrix LCDs. The images are saved in the buffer and the screen refresh is done every 60Hz. Electronic devices generate the inner light of LCDs, in order to avoid the total dependency of external luminous sources.

It is possible to display colours by using different materials or dyes and placing a set of three coloured sub pixels in every screen position. More information about LCDs in [9, 11, 12].

Another construction method of LCDs consists by placing a transistor in every pixel position. This technology is known as Thin Film Transistor (TFT). Transistors control the voltage and prevent the gradually charge leak from liquid crystal cells. These devices are called active matrix displays. To get more information about active matrix displays see [12] and for TFT technology visit [13].

a) *Calculator* ***b)*** *Laptop*

Figure 2.2.6 Examples of LCD devices.

2.2.2 Image Storage

The graphic programs used in image storage are divided in two classes, raster and vector. Both programs are described in following sections.

2.2.2.1 Raster Programs

The raster refers to the region of the monitor (CRT or LCD) capable of rendering the images. It is a grid with “x” and “y” coordinates in a screen. If it is necessary to work with three-dimensional pictures, the “z” coordinate is added.

To create images in a CRT monitor, an electron beam covers the screen from left to right and from top to bottom. It is very similar to what happens to televisions, with the slight difference that in TVs the image is interlaced, to wit, odd lines are projected first and then the even lines. This is possible to do because the human eye cannot perceive the change from odd lines to even lines. More information about TV image transmission is located at [6]. The generated grid (raster) is slightly smaller than the full screen size of the monitor. The advantage these monitors offer is that the height and width are adjustable, as well as the horizontal and vertical position.

In LCDs the raster is known as grid and its obtaining procedure is different from the CRTs. The image elements are represented individually instead of dividing the screen into lines. For these devices it is not possible to modify the grid. For instance, only part of the screen is covered with the displayed image when choosing a resolution of 640x480 pixels in a LCD intended for 800x600 pixels. On the other hand, setting a resolution of 1024x768 in the same 800x600 LCD, the image exceeds the screen area and scrolling is necessary to see the entire image.

Raster programs work with pixels and the resulting format is known as bitmap. The program divides the image in tiny dots, whose size do not exceed 1/300 of square inch, and stores the information of each of them. For example, a square inch with a resolution of 300 dots per inch (dpi) needs 90,000 dots. If the picture is black and white, only one bit is needed to represent each image dot. Talking about chromatic pictures or gray scale pictures, two or more bits are needed to represent each dot.

Gray scale images have generally 16 or 256 different gray shades; correspondingly every dot uses 4 or 8 bits. Colour pictures have a range that goes from 16 colours (4 bits per dot) up to 16.7 millions of colours (24 bits per dot). Although it is possible to use 32 bits per dot (4,300 millions of colours approximately), 24 bits are taken as a maximum because the human eye is not able to notice the differences between both of them. The required amount of bits to represent the colour of an image dot is known as image colour depth. For more information about raster programs see [9].

2.2.2.2 Vector Programs

These programs have lines or vectors as basic element. The vectors are used as a repetitive pattern to create objects and shapes. To create images in these programs it is necessary to work with commands or mathematical instructions, which place the lines and figures in a given two-dimensional or three-dimensional space.

In physics, vectors represent quantity and direction at the same time. In these programs, the graphic files are saved as a sequence of vectors. In raster programs, the file contains one bit for each image dot. On the other hand, in vector graphics, each vector represents a row of connected dots, which leads to a reduction of the file size.

Vector files are sometimes called geometric files. The most known examples of this family are the computer assisted design programs (CAD) and map generators. Animated images are generally vector files. For instance, Shockwave's Flash™ creates 2D and 3D animations that travel across the Internet as vector files and become rasterized when arriving at the screen.

It is difficult to say which programs are better; it depends on the applications they are required for. Raster programs generate big files and it is complicated to modify them without loosing image information. There are programs that transform raster files into vector files and *viceversa*. Thanks to this, image modifications are possible without data loss. Raster files are easier to transport between compatible systems. Each time a scanner or a web cam is used, a raster image is generated automatically.

2.2.3 Image Compression

The value of image compressions is better explained by the following example. consider a 5"x5" image with 256 colours and a resolution of 300 dpi. The picture has 25 square inches and 90,000 bits per square inch. Multiplying inches by resolution result in 2,250,000 bits needed to display the image. Adding 8 bits to represent colour depth, a minimum of 2.25MB of disk space is needed for this small image. It is minimum because the file has a header containing file information like size, resolution and colour depth.

This gives an idea of how much disk space images need, in addition remains the problem of sending these files over Internet. For instance, with a 56Kbps modem the transfer of the last 2.25MB image takes more than 5 minutes.

Image compression refers to the bytes reduction of a graphic file. This reduction allows a better use of disk space. There are two types of image compression in Computer Science, lossy and lossless. With the lossless technique, every information bit of the file is restored when the decompression is made. Lossless compression is widely used in text files (data sheets, program code, etc.) since information loss would provoke serious problems. The most known graphic formats that use this technique are GIF™ and PNG™.

On the other hand we have the lossy compression, which reduces the file size by eliminating certain generally redundant information. When the file is decompressed, only part of the original date is present, even if the user cannot notice it. This technique is used normally in audio and video applications, where most users do not detect the data loss. The main example of lossy compression is the JPEG™ format. A brief introduction to these formats is given in following sections.

2.2.3.1 GIF

The name comes from Graphic Interchange Format. This format is one of the preferred for image handling on Internet. It uses the LZW compression algorithm, developed by Abraham Lempel, Jacob Ziv and Terry Welch. This algorithm is also used in text compression. Its *modus operandi* is as follows: it takes a sequence of bits of specific length, 10 bits for example, and it creates an entry table for this characters pattern, which consists of the pattern itself and additional code. When reading further information, the saved patterns are substituted by the mentioned short code, generating this way a smaller size file. For more information about LZW compression algorithm see [14].

GIF format uses two-dimensional raster data type and is codified in binary. There are two versions of the format, 87a and 89a. The last version allows the display of animated GIFs, which are a sequence of images in a single file. This format is widely used on Internet for pop up advertising and banners, thanks to their small size. GIF89a format can also display interlaced GIFs.

Interlaced GIF is an image that appears on the screen as if it materializes itself. At the beginning only the diffuse outline of the image is displayed, but it is replaced by seven successive data waves that fill the holes in the lines that are not yet shown. The process continues until the image is displayed as a whole. Among the advantages that this format offers to the users with low-speed Internet connection (for example 14.4Kbps and 28.8Kbps modems) is that the image appears to be transmitted in a faster way, and some times it is not necessary to wait until the whole image is charged to have an idea of the content. For users with high-speed Internet connections, the difference between normal GIF and interlaced GIF is almost imperceptible.

The GIF palette works only with 256 colours, for this reason it is convenient to use this format when the image to display has solid colours or defined blocks without shadows. Its compression condenses a group of bits with the same colour in a single instruction line, for instance 58 green bits.

Another property of GIF format is the generation of transparences. When saving the file, a transparent colour in the image can be defined and when placing the image over different backgrounds, a higher versatility of image displaying is acquired. The next figure shows an example of transparence. More information about this format can be found in [9, 15, 16, 17].

a) *Original GIF* ***b)*** *Transparence*

Figure 2.2.7 Examples of GIF format.

2.2.3.2 PNG

Portable Network Graphics, This format, like GIF, is lossless, and as an additional advantage, its license is free. PNG compresses the images between 5% and 25% more than GIF format. It is believed that in a near future PNG will replace GIF. Some of its characteristics are:

- Not only accepts colour transparency, but the degree of transparency can be also adjusted.
- Interlacing is faster than in GIF.
- Images can be saved with true colour (2^{24} bits), in gray scale as well as normal palette.

The advantage that still has GIF over PNG is that the latter does not support animations, because PNG cannot save multiple images in one file. To get more information about PNG format see [16, 17, 18]. Figure 2.2.8 shows an example of PNG, it is noticeable the smooth colour variations that can be handled.

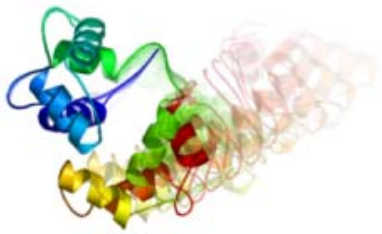


Figure 2.2.8 PNG image.

2.2.3.3 JPEG

Its name comes from the abbreviation of Joint Photographic Experts Group. This association, which belongs to the International Organization for Standardization (ISO) [19], develops compression algorithms for graphic files.

This format is lossy, since the image quality can be specified; it depends on the type of compression that is made. The lower is the compression, the better results the image quality, but also the image size will be bigger. There is a balance between file size and image quality.

JPEG format is normally used when the image has different shades or shadows, for example, photographs, thanks to its capacity of supporting true colour (2^{24} bits, 16.7 millions of colours). The figure 2.2.9 is a representative example of the advantages JPEG offers when working with complex images that contain diffuse tones and shadows.

However, JPEG does not work very well with simple images based in lines, blocks without shadows or slight colour graduations, due to its lossy compression technique, which produces a worse image resolution. In addition, the size reduction of this type of images is lower compared with GIF or PNG formats.

Figure 2.2.9 JPEG landscape.

There is also another JPEG format, the progressive JPEG, which blurs the image slowly until it appears as a whole. It works the same way as interlaced GIF. As mentioned, this format is useful for users with low speed modems. JPEG along with GIF and PNG are the most used in Internet. More information about JPEG format, in [16, 17, 20].

It is impossible to tell which format is better than the other ones because their applications or action fields are not the same; PNG manages to get a higher compression than GIF, but it is not able to handle animations. JPEG is the best option for photographs due to its shadows that smooth the image, allowing not loosing quality (at least perceptible) by compressing the file, although some information is lost during the process.

On the other hand JPEG is not recommended when using vector images or simple traces; the most suitable for these cases is the GIF format. These are the three most used formats in Internet due their small size and portability. For more information and tips of using those formats and web pages design see [16].

2.3 Colour Models

Colour is not an intrinsic property of objects, but it depends on the light that falls on them. Visible light is only a small section of electromagnetic spectrum that humans are able to detect by the sense of sight. A representation of the electromagnetic spectrum is shown in figure 2.3.1. The spectrum is divided in frequencies (Hertz) or in wavelengths (λ), specifying the different components. They are radio waves AM and FM, television, microwaves, infrared light, ultraviolet light, X rays, cosmic rays and visible light.

The range of seeing for humans starts around 630 nanometers (nm), which is red colour. As the wavelength diminishes, oranges and yellows are found in the spectrum. After that, come green and blue tones and finally violet, which has approximately 400nm of wavelength. There are people able to notice more wavelengths that the ones in the visible light, some of them can see part of the infrared light and some other can detect ultraviolet light hues.

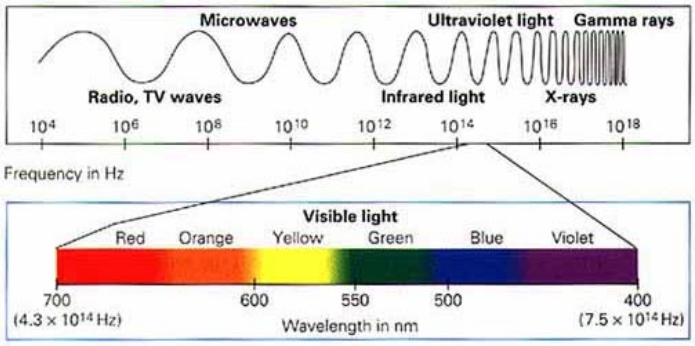


Figure 2.3.1 Electromagnetic spectrum

The sun emits white light, to wit, the union of all colours, but depending of the object it illuminates, the colour changes, because some waves are absorbed and other are reflected by that object. The combination of the reflected waves is the colour that is perceived. This property is called hue or colour of light.

An important light characteristic is the luminance, that is, the luminous intensity of a surface in a given direction per unit of projected area. The third light property is the saturation or purity. Saturation tells how pale or solid is the colour. Pastel and pale colours are less pure. These three characteristics, colour, luminance and purity are used to describe different light properties. The term chromaticity refers to purity and wavelength of colour.

When an artist works with colours, he obtains different tones, shadows and hues when adding white or black to the main tint. By adding black pigments, shadows are generated, the bigger is the amount of black, and the darker gets the shadow of the colour.

By adding white, hues are obtained. If white and black pigments are added, different tones are obtained. For more information about the human interpretation of colour, history and glossary about colour terms, see [21].

2.3.1 Munsell Colour System

Albert Munsell designed his own system that measures, specifies and associates colours. In Munsell system each colour has three attributes, hue, value and chroma. He established numerical scales with visible uniform graduations for each of these qualities.

Hue is the characteristic that allows distinguishing one colour from another, for example, red from green or from yellow, etc. There is a natural order of hues and it is represented in the electromagnetic spectrum, to wit, red, yellow, green, blue and purple. By mixing adjacent colours, medium tones are obtained, for example, the mixture of red and yellow results in orange. The remaining combinations are, yellow-green, green-blue, blue-purple and purple-red. This series returns to the starting point, which is the reason why Munsell represented the hue as a circle. There are ten different hues in this model, the five mains and the combinations between adjacent tones. He abbreviated them by taking the first letter of each colour, obtaining R, RY, Y, YG, G, GB, B, BP, P, and PR. Figure 2.3.2a shows the hue circle.

Value is the quality that measures the colour lightness. Its range goes from 0 (pure black) up to 10 (pure white). The medium gray tones are known as neutral colours, which lack hue. Colours that contain hue are called chromatic. The scale of value applies for both, neutral and chromatic.

The last parameter of Munsell categorization is the chroma of colour. When a colour has low chroma is weak and if its chroma is high, it is called highly saturated. For instance, consider the mixture of light yellow and gray. Starting on gray and adding yellow gradually until a vivid yellow is obtained, a colour series with different values is displayed, being gray (neutral) the less saturated and vivid yellow, the one with highest chroma.

Value scale starts in zero, but it does not have a defined maximum. It depends on the specified colours. Normal colours can surpass 20 units. Some fluorescent colours exceed the mark of 30 units. The next figure illustrates the three attributes of colour according to Munsell.

a) *Hue* ***b)*** *Value* ***c)*** *Chroma*

Figure 2.3.2 Munsell colour attributes.

Munsell notation for chromatic colours is defined as follows: H V/C, where H is hue, V is value and C is chroma. For a vivid red with a hue of 6, value of 7 and chroma of 16, its notation is 7R 8/17. It is allowed also the use of decimals if they are required, for instance, 7.3R 8.4/16.8. The notation for neutral colours is: N V/ (chroma of neutral colours is zero, but it is possible to omit it). The notation N 1/ is black, N 5/ gray and N 10/ white.

Each quality of Munsell colour can be altered independently and the model can be set in a three-dimensional space. Neutral colours are located on the vertical axis, which is called neutral axis. Black is on the low part and white on the top of this axis. Hues are located around the neutral axis. The chroma is perpendicular to vertical axis and increases to the ends. This three-dimensional arrangement is known as Munsell colour space. In figure 2.3.3 this space is represented. For more information about this theme, see [22, 23].

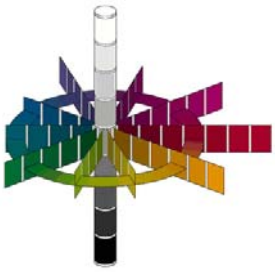


Figure 2.3.3 Munsell colour space.

2.3.2 XYZ Model of the CIE

Due to the unfeasibility to display all possible colour by combining a finite set of luminous source, in 1931, the *Commission Internationale de L'Eclairage* (CIE) defined three standard primary colours. These three standard primaries are imaginary colours and are defined mathematically with functions that allow matching positive colours that specify the quantity of each primary colour needed to describe any other colour from the spectrum. After this analysis, an international standard definition for every colour was obtained.

The set of primary colours of CIE is known as the XYZ colour model, where “X”, “Y” and “Z” represent vectors in a three-dimensional space of additive colour. Every specific colour “C” can be reproduced with the equation “C = xX + yY + zZ”, where “x”, “y” and “z” are coefficients of primary colours required to represent the desired colour “C”. It is convenient to normalize the equation quantities with the luminance (X+Y+Z). The normalized coefficients are obtained as follows

$$x = \frac{X}{X+Y+Z}, \qquad y = \frac{Y}{X+Y+Z}, \qquad z = \frac{Z}{X+Y+Z}$$

where “x”+“y”+“z”=1. So any colour can be represented in function of x and y. These normalized values x and y are called chromaticity, due their dependence to hue and purity. If colours are specified just in function of “x” and “y”, “X”, “Y” and “Z” cannot be achieved.

A more complete description of a colour is reached with the values of “x”, “y” and “Y”. The remaining quantities can be calculated in this way

$$X = \frac{x}{y} Y, \qquad Z = \frac{z}{y} Y$$

where “z”=1–“x”–“y”. By using the chromaticity coordinates (“x”, “y”) it is feasible to represent all colours in a two-dimensional graphic. This representation is known as the chromaticity diagram of the CIE. Figure 2.3.4 shows this diagram.

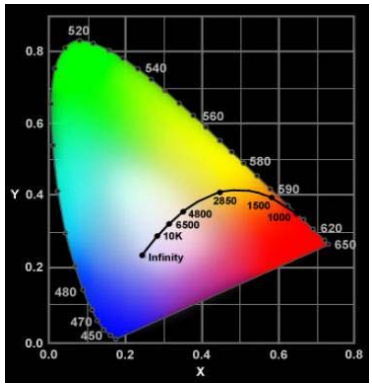


Figure 2.3.4 Chromaticity diagram. Colour positions are defined in nanometers.

Dots lengthwise the curve are the pure colours of the electromagnetic spectrum, defined according to their wavelength (nanometers) from the red extreme to violet extreme. The inner dots represent all possible combinations that can be visualized in a cathode ray tube (CRT) by mixing red, green and blue. The middle of the graphic shows the union of all colours, the white light.

CIE diagram is a useful tool for colour definition; its main operations are the gamma comparison for different primary sets, complementary colour identification and obtaining wavelengths and dominant purity of a specific colour. However this system is not based on the human colour perception, or in hardware devices. More information about the CIE can be obtained in [24, 25].

There are also models based on colour display; they are in charge of representing colour in different devices like computers, televisions and printers. These models are RGB, CMY and YIQ/YUV. RGB and CMY models have great application in computers. YIQ model is the pattern to follow in televisions.

2.3.3 RGB Model

The name is given by the abbreviation of Red, Green and Blue, the main hues that are perceived by the cone cells of human eye.

These colours have a peak sensibility in the wavelengths around 630nm (red), 530nm (green) and 450nm (blue). These three components are the base of monitor displays; with them an orthogonal vector system is built. The null vector (0,0,0) represents black; vector (1,1,1) corresponds to white; (0.5,0.5,0.5) is gray; red is the vector (1,0,0), green (0,1,0) and blue (0,0,1). The remaining colours are obtained by the combination of red, green and blue hues. At the main diagonal is located the gray scale also known as monochromatic scale. Figure 2.3.5 explains the previous description.

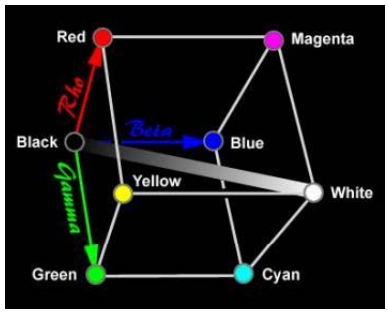


Figure 2.3.5 RGB cube.

This is an additive system. By adding the primary colour intensities, different hues can be achieved. Each colour can be represented as a three coordinate's vector (R,G,B) in the colour plane and its coordinate values go from 0 to 1. For instance, a colour "C", expressed with RGB components, will be "C=xR+yG+zB", where "x", "y" and "z" are coefficients between 0 and 1.

The RGB system is easy to build and has a direct relationship with hardware; due to video cameras give amplitude and continuous signal for the three components. In this model, each pixel (picture element) on screen has a RGB value. A specific bit number is necessary for each pixel to transfer colour information. To represent all the colours that human is able to recognize, every RGB component is defined with eight bits. This results in 2^{24} (16,777,216) different colours that can be obtained. For more information about RGB, see [25, 26, 27].

2.3.4 CMY Model

This model is used mainly in printers. Its colours Cyan, Magenta and Yellow are the complementary colours of Red, Green and Blue respectively. Video monitors generate the colours by projecting an electron beam on the screen coated with phosphors (red, green and blue); that is the reason why monitors use the RGB model. Printers work in a different way. The tracers create various tones by covering the paper with colour pigments, that is, in the paper, the light is reflected and the colours are displayed in a subtractive process.

For example, the mixture of green and red forms yellow, so when a yellow printing reflects the white light, this must not contain any blue pigment. Yellow ink subtracts blue light, cyan ink absorbs the light red component and magenta ink does the same with green light [28].

The CMY cube is identical to the RGB cube but the vertex origin (0,0,0) in CMY is the white light, while the opposite extreme (1,1,1) represents black. The rest of colours are obtained in a subtractive way, for example, green results by combining yellow and cyan.

The printing process used in CMY model generates a coloured dot when four ink dots are combined; an analog process to RGB, but RGB displays a coloured dot by combining three phosphor dots. CMY requires one dot for every primary colour and one black dot. It is necessary to add black ink because the combination of cyan, magenta and yellow gives dark gray instead of black.

These models are very close to each other; therefore, the transition from one system to another is simple. It is calculated as a matrix operation as follows

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{and vice versa} \quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}.$$

Going from CMY to RGB, the unitary vector is white. In the opposite direction, black is represented as the vector (1,1,1).

RGB model is optimum to display colours in a computer monitor, but working with television requires another way to handle colours. The colour model of National Television System Committee (NTSC) is the YIQ or YUV.

2.3.5 YIQ Model

This model, instead of separating the colour in red, green and blue, it divides it in intensity or luminance “Y”, which is basically black and white information and two colour elements “U”, “V” or “I”, “Q”, known as chromaticity. For the “Y” parameter a combination of red, green and blue intensities is selected to represent the luminance standard curve.

Black and white televisions only use the intensity “Y”, while colour displays use both luminance and chrominance information. The “Y” parameter is assigned to the widest bandwidth on the NTSC video signal frequency (4 MHz approximately). The “I” component transmits the information of orange and cyan hues, which correspond to the shadows of the skin colour. Its broadband is located near 1.5MHz. The “Q” parameter holds the greens and magentas information in a broadband of 0.6MHz. More information about this model in [29].

The YIQ model fits better with the human vision, since human eyes deal with brightness and colour. Actually this model works in an analog way to what happens in human eye with colour photoreceptors (cones) and brightness cells (rods) [3]. The components “Q” and “I” are selected because of hardware compatibility used in transmission. The “I” signal is acquired by subtracting cyan to red, while “Q” is magenta minus green. The next table shows the conversion from RGB to YIQ.

Table 2.3.1 Conversion from RGB to YIQ.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{vmatrix} 0.299 & 0.587 & 0.144 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.528 & 0.311 \end{vmatrix} \bullet \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

To convert a NTSC signal to RGB, is required a decoder that separates the video signal in the YIQ components and after that converts them to RGB values. Table 3.2 shows the inverse matrix operation from YIQ to RGB.

Table 2.3.2 Conversion from YIQ to RGB.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{vmatrix} 1.000 & 0.956 & 0.620 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.108 & 1.705 \end{vmatrix} \bullet \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

There are also some systems based on the human perception of colour. These are HSV model (Hue, Saturation and Value), HLS model (Hue, Light and Saturation) and HIS model (Hue, Saturation and Intensity).

Hue is the colour defined by its wavelength, it distinguishes from red, yellow, violet, etc. Saturation is the quantity of present colour, it differences between pink and red. Value is the colour strength; it indicates how pale or pure the colour is. Light, brightness or clarity refers to the luminous intensity emitted by the colour; it perceives the difference between light blue and dark blue.

2.3.6 HSV Model

The HSV model, created in 1978 by Alvey Ray Smith, is very similar to Munsell colour space, used by designers, artist and manufacturers. The colour parameters for this system are hue "H", saturation "S" and value "V". The three-dimensional representation of the HSV model is extracted from the RGB cube. When observing the RGB cube along the main diagonal that goes from white to black, the cube contour seems like a hexagon, see figure 2.3.6. The border of the HSV hexcone contains the different hues. Saturation "S" is measured along a horizontal axis and value "V" is given along the vertical axis that crosses the pyramid in the middle.

The hexcone hues have a separation of 60 degrees from each other. Red is at the origin, yellow in 60°, green in 120°, cyan in 180°, blue in 240° and magenta in 300°. Complementary colours are separated by 180 degrees.

Saturation goes from 0 to 1 and represents the purity of a selected hue, being S=1 the maximal purity to achieve. When S=0, the resulting colour is gray. Value varies from 0 on the pyramid apex (black) to 1 on the top part. Colours have their maximum intensity on the top of the prism. When S=1 and V=1, pure hues are acquired. White is at the point S=0 and V=1.

This is an intuitive model, it starts with the selection of a pure colour by choosing an angle at the hexcone and setting S=V=1. By the addition of black, V will decrease but S will remain constant. For instance, a dark green has S=1, V=0.3 and H=120°. Adding white to the colour diminishes the saturation but it does not affect the value. By adding black and white, both value and saturation are reduced.

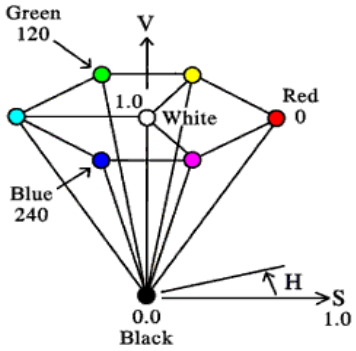


Figure 2.3.6 HSV Hexcone

Colour concepts that are related to the terms shadows, tones, shades, are presented in a transversal plane of the HSV pyramid, see figure 2.3.7. By adding white to a pure hue, different tones on the top part of the hexcone are generated, where the parameter coefficients are V=1 and $0 \leq S \leq 1$.

If black is added to a pure hue, "V" is reduced on the pyramid side. The parameter values are S=1 and $0 \leq V \leq 1$. Several tones result of the addition of black and white and also colour dots on the triangular transversal area of the pyramid are generated.

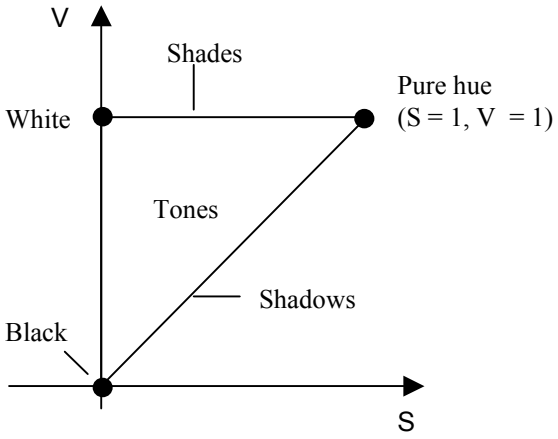


Figure 2.3.7 Transversal section of the hexcone that represents shadows, tones and shades.

Human eye can identify around 128 different tints and more or less 130 levels of saturation (shades). For each of them, a number of shadows can be recognized (value grades), depending on the selected tint. It is possible to detect near to 23 shadows in yellows and approximately 16 different shadows in the blue part of the spectrum.

By multiplying parameters, the number of colours that humans can perceive is obtained, to wit, 128x130x23=382,720. For most applications, 128 tints, 8 levels of saturation and 15 value specifications are enough. With this range of parameters, the user works with 16,348 colours and the system requires 14 bits of disk space per pixel. More references about this model, in [25, 30].

2.3.7 HLS Model

In 1978 Tektronik™ developed the HLS model and it was wide used in chromatic computers at the end of seventies. This model, like the HSV, is based on intuitive colour parameters. The representation of this model is a double hexcone and its three attributes are hue "H", lightness "L" and saturation "S". Figure 2.3.8 displays the HLS double hexcone. Hue works the same way as in HSV model. It specifies an angle around the horizontal axis that locates the selected hue. Complementary colours are separated by 180 degrees.

The vertical axis in this model is the lightness "L". Black is located in L=0 and white in L=1, having as intermediate dots the gray tones. The pure hues are in L=0.5. Saturation defines the relative purity of colour. This attribute goes from 0 to 1 and the pure hues are the ones with S=1 and L=0.5. As "S" diminishes, the hues become less pure. S=0 represents the gray scale.

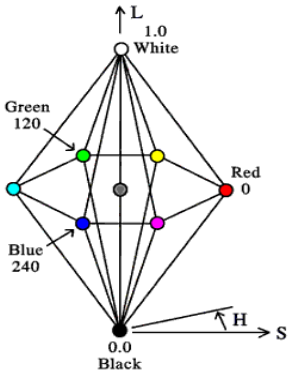


Figure 2.3.8 HLS double hexcone

As mentioned, this model is intuitive. As basis is selected a hue "H" and lightness "L" as well as saturation "S" are increased or diminished to obtain the desired colour. Incrementing the lightness, the colours become clearer and they get darker by decreasing L. When the saturation is reduced, colours approach to gray. More information about this model can be found at [25, 31].

There is another model called HSI, which is the same as HLS but with some differences. "H" is the hue attribute, "S" is saturation and "I" is intensity, while in HLS is known as lightness "L". This model is a double hexcone identical to the HLS, only that its vertical axis is the intensity. To make a conversion from RGB to HSI the next table is used. To know more about HSI model, see [32].

Table 2.3.3 Conversion from RGB to HIS.

$\mathbf{H} = [\pi/2 - \arctan \{(2 \cdot \mathbf{R} - \mathbf{G} - \mathbf{B}) / \sqrt{3} (\mathbf{G} - \mathbf{B})\} + \pi; \mathbf{G} < \mathbf{B}] / 2\pi$

$\mathbf{I} = (\mathbf{R} + \mathbf{G} + \mathbf{B}) / 3$

$\mathbf{S} = 1 - [\min (\mathbf{R}, \mathbf{G}, \mathbf{B}) / \mathbf{I}]$

3 Textures

What is a texture? Texture is a basic property of an image, like colour or shape; however, there are no formal definitions about it. Looking at Webster Dictionary ™, the following concepts are found "*a) something composed of closely interwoven elements; b) the structure formed by the threads of a fabric*". The etymology given also by Webster Dictionary ™ says "*from Latin textura, from textus, past participle of texere to weave*", which leads to the conclusion that the word "texture", describes a regular pattern from which a whole is made.

Speaking of Image Processing, textures are a grey distribution of an image surface. There are different types of textures, and they are classified according to the complexity of the pattern, although there is not an official categorization. Usually the textures are catalogued in these groups:

- Strong ordered textures or structural textures:
  These are composed of elemental patterns which are organized with a specific arrangement. Some examples are bricks, reptile skin or grid.

- Random textures:
  These textures are created stochastically. They are often found in the nature, like grass, rain, snow. Using fractals, some of these textures can be constructed, however the artificially created will be slightly different than the originals; hence the fractals are not as accidental as it is Nature.

- Directional textures:
  These types of textures are the ones, where the pixels possess a local orientation or direction, for example, water or wood.

Texture analysis is one of the branches of Image Processing that does not have solid basis; many different empiric and semi-empiric analysis can be found in the literature referent to textures. In this work, the observed textures are mostly directional or strong ordered.

Texture analysis field of action is very wide, some of the most known applications are remote sensing, automated inspection, and medical image processing. Texture analysis can be used to find the texture boundaries, called texture segmentation. Texture analysis can be helpful when objects in an image are more characterized by their texture than by intensity, and traditional thresholding techniques cannot be used effectively. For general information about textures see [33, 34, 35].

3.1 Approaches

There are two types of statistical approaches used for textures, first order methods and second order methods. In the first group, histograms are the most common and used to obtain texture information. The maximum exponent of the second group is the co-occurrence matrix.

3.1.1 Histograms

Histograms are statistical methods, which help to get a more precise idea of the image colour distribution. Histograms are, generally, two dimensional graphics that display an independent variable on the horizontal axis, and the dependent variable along the vertical axis.

Histograms are first order methods that tell how many pixels correspond to a given colour value in an image. It is called of first order because single pixels are analysed, without considering the relationship with neighbour pixels. Normally, images of 8-bit data are used; therefore histograms have a range of 256 possible grey values from 0 (black) to 255 (white).

Obviously this cannot be reduced to black and white images, but also to colour images, therefore histograms can also represent other shades or tones, for example, red, green and blue scales, which are really important to consider in computer graphics, since this is the colour model to follow, that is, RGB model.

If the image itself is already in black and white and red, green and blue histograms are calculated, they throw the same result as the greyscale histogram. By taking the same amount of red, green and blue, the obtained colour travels always across the main diagonal of the RGB cube, *videlicet*, the greyscale, as shown in next figure, the RGB cube.

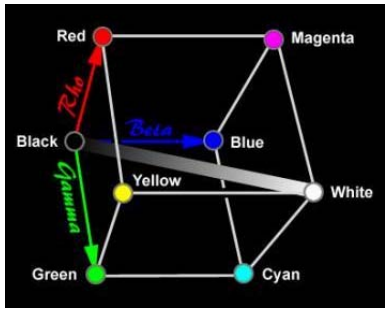


Figure 3.1.1 RGB cube.

The histogram "h" of an image "p" is calculated with the following function

$$h(v) = \frac{1}{MxN}\left(\sum_{\forall i,j:p(i,j)=v} 1\right)$$

Where "v" is the number of colour shades (red, green, blue or grey), "M" and "N" are width and height of the image "p", and finally "i" and "j" are the index of the coordinate to locate the pixels in the image. To calculate the red, green and blue histograms, the process extracts the desired colour of the image and takes it for the histogram. For the grey scale it is necessary to apply this equation:

Grey value = 0.33*red value +0.56*green value +0.11*blue value.

As observed, it is not simply summing the three values and dividing by three, since the human eye is sensitive in different degrees to red, green and blue shades. Figure 3.3.2 shows examples of the red, green and blue histograms, as well as the combined greyscale histogram taken from a beach image to appreciate the difference in spectra of the colour channels.

These histograms differ from each other in this particular image, due to the fact that the amount of red, green and blue is quite different along the picture. More information about histograms can be found in [36, 37, 38] and colour histograms in [39].

Figure 3.3.2 Colour histograms. a) red channel, b) green channel, c) blue channel and d) grey channel (sum of the last three channels).

Using histograms, another important characteristic of the image can be obtained, the entropy. Entropy is a measure of chaos contained in the image, following, its equation:

$$H = -\sum_{v=0}^{255} h(v) \cdot \log_2 h(v)$$

Where "h" is the histogram and "v" is the colour value (red, green, blue or grey). Further explanations of entropy in image processing can be found in [40].

3.1.2 Grey Level Co-occurrence Matrix

The characterization of a texture with a histogram is not enough, so Haralick, in 1973, designed a second order analysis for a better analysis of textures, this was the grey level co-occurrence matrix. It is called of second order, because for this analysis, the neighbour pixel behaviour is fundamental. Co-occurrence matrices calculate how often grey values appear at two pixels separated by a given distance and direction.

To create a grey level co-occurrence (GLCM) a *reference pixel* and a *neighbour pixel* are needed, the relationship between them is called *direction* or *spatial relation* and refers to values North, South, East, West and the diagonals, NE, NW, SE, SW, which can also be expressed in coordinates, for example, (-1,0) represent the West direction, that is, one pixel to the left of the reference pixel and no movement in the vertical axis, (1,0) is the East direction and so on. Spatial relation can be also represented as a degree function, see figure 3.3.3.

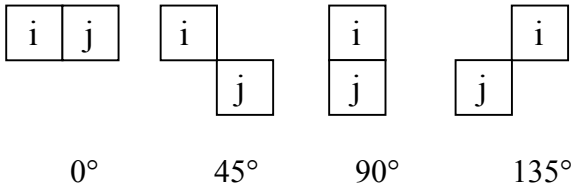


Figure 3.3.3 Degree function representation of the orientation of co-occurrences.

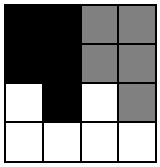
The properties of a GLCM are:

a) It is square. The reference pixels and the neighbour pixels have the same range of values.

b) The number of rows and columns is decided by the depth level of the image. An 8-bit data image has 256 possible values that result in 256x256 square matrix with 65,536 cells. 16-bit data give a 65536x65536 square matrix with 4,294,967,296 cells, which is already very difficult to handle, considering the time of calculation.

c) It is symmetrical along the diagonal. Texture calculations are better performed with symmetric matrices. Defining a spatial relation to East, does not result in a symmetric matrix, but the West matrix can be obtained by transposing the East matrix, *videlicet*, interchanging rows and columns. Adding both of them results in a horizontal matrix, which is symmetric.

More information about GLCM is located in [41, 42]. For a better explanation of the GLCM, an example is presented. Consider an image with 3 different grey levels (black, white and grey) and its numeric representation, see figure 3.3.4.



a)

0	0	1	1
0	0	1	1
2	0	2	1
2	2	2	2

b)

Figure 3.3.4 Three grey levels image. a) colour representation, b) numerical representation.

Calculating a West direction (-1,0) the following GLCM is obtained:

	0	**1**	**2**
0	2	0	1
1	2	2	1
2	1	0	3

This is not a symmetric matrix, but calculating the transpose (East matrix) and adding it to the West matrix, a symmetric horizontal matrix is obtained.

West matrix

	0	**1**	**2**
0	2	2	1
1	0	2	0
2	1	1	3

East matrix

	0	**1**	**2**
0	2	0	1
1	2	2	1
2	1	0	3

Horizontal matrix

	0	**1**	**2**
0	4	2	2
1	2	4	1
2	2	1	6

Now it only remains to express the GLCM as a probability, this process is called normalisation. This is the equation for the normalisation:

$$P_{i,j} = \frac{V_{i,j}}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} V_{i,j}}$$

Where "P" is the normalized pixel, "i" and "j" represent the row and column index respectively, "V" is the value of the pixel and "N" is the number of grey values. In previous example, the values are divided by 24 to obtain the normalised GLCM:

	0	**1**	**2**
0	0.1667	0.0833	0.0833
1	0.0833	0.1667	0.0416
2	0.0833	0.0416	0.25

The calculations based on the normalized GLMC are:

Contrast:

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(i-j)^2 P_{i,j}$$

This feature is a measure of the clearness with which different regions of the image are visible. The elements along the main diagonal are not considered while calculating. The image has a low level of contrast if the majority of the values are located along the main diagonal. The images with low contrast have a high amount of neighbour pixels with almost the same grey value; in this case, "i" is very similar to "j". On the other hand, images with high contrast level have the values dispersed from the main diagonal.

Homogeneity:

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\frac{P_{i,j}}{1+|i-j|}$$

The homogeneity can be taken as inversely proportional to the contrast, due this feature calculates the smoothness in an image. The grey levels vary slowly across the image.

Entropy:

$$-\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}P_{i,j}\log_2 P_{i,j}$$

It is a measure of chaos in the texture. It represents how much randomness is contained in the image. A texture with only one grey shade has zero entropy, since is uniform. A texture is orderless when the occurrence values are big but located in just a few cells of the matrix.

Energy:

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(P_{i,j})^2$$

Energy is the opposite of entropy. This feature measures the homogeneity of a surface or how orderly is the texture.

Autocorrelation:

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(i+1)\cdot(j+1)\cdot P_{i,j}$$

Thanks to this property, the elements with large values are heightened. This feature expresses in which direction is the matrix ordered, if it tends to white (both “i” and “j” large or to black (“i” and “j” small). It also shows the direction of the texture, it means, if it tends to be vertical, horizontal or else.

These properties are not exclusive for black and white textures. If the image information is separated in its three colour components (red, green and blue), co-occurrences matrices can be obtained for each colour channel, allowing the analysis and manipulation of chromatic textures. For more information about co-occurrence matrix based analysis, see [43].

4 Stochastic Learning of Textures

4.1 Introduction

In this chapter, two methods are used for the learning of textures. They acquire the image information in a completely different way. However, when reconstructing the image, they follow the same principle.

The first experimental part of this work is done using co-occurrence matrices [41, 42, 43]; grey level co-occurrence matrix is not considered, instead of it, the red, green and blue channels are calculated to acquire the picture information. Once the image is stored, the process creates distribution curves for each colour channel. The selection of the appropriate colour shade is randomly determined by the reconstruction function.

The second method used for the learning process is the vector quantization. The vector components handle also the three colour channels, allowing the manipulation and analysis of chromatic samples. The initial vectors are selected randomly, and they are responsible for the obtaining of the image information. The next step, like in the co-occurrence matrices method, creates distribution functions for each colour channel. The final stage is the colour shade selection, which is randomly calculated.

4.2 Learning based on Co-occurrence Matrices

4.2.1 First approach: Small Working Area of Co-occurrences

4.2.1.1 Acquiring information

For the storing of the obtained information from the textures, three matrices are needed (one for each colour). A working area scans the original sample and extracts its colour components to store them in the correspondent co-occurrence matrices. The first step defines this working area, for instance 3x3 pixels, 4x2 pixels, etc. The main matrices are defined like:

```
float smred [ver][hor][256][256];
float smgreen [ver][hor] [256][256];
float smblue [ver][hor] [256][256];
```

Where “ver” and “hor” correspond to the size of the working area and the next parameters represent the range of values for each colour channel, namely 0 to 255. Image co-occurrence values are established by the three colour channels, which are obtained as this area scans the image. The direction of this co-occurrence is defined by the working area, in this case, the lower right quadrant, like illustrated in figure 4.2.1.

Next step scans the image and extract, from each pixel, the three colour components of the image separately and calculate the co-occurrence matrices. Algorithm 4.2.1 shows this procedure.

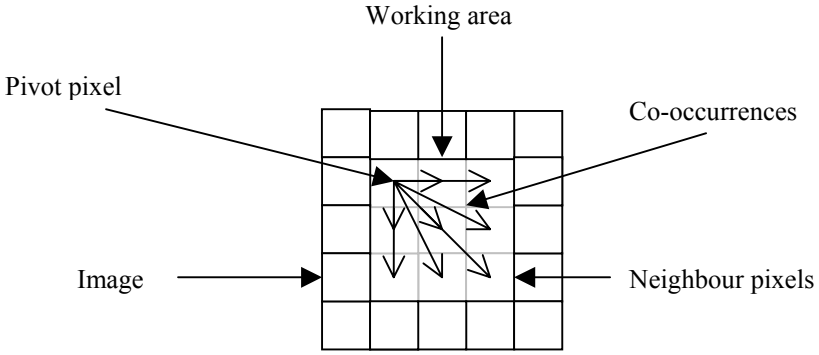


Figure 4.2.1 Example of the working area getting information of the image.

Algorithm 4.2.1 Obtaining of co-occurrence matrices.

```
For i=0 to ver
 For j=0 to hor {
  if (i<ver-1) or (j < hor-1)
   disti=ver-1-i; distj=hor-1-j;
   For pi=0 to Imageheight - disti
    For pj=0 to Imagewidtht- distj
      smred[i][j][red of Image(pi, pj)] [red of Image(pi+disti, pj+distj)] +=1;
      smgreen[i][j][green of Image(pi, pj)] [green of Image(pi+disti, pj+distj)] +=1;
      smblue[i][j] [blue of Image(pi, pj)] [blue of Image(pi+disti, pj+distj)] +=1;
```

This process scans the image with the working area and gets the number of occurrences as travels along the image. The main loops are defined by the variables “ver” and “hor”, *videlicet*, the working area.

In fact, a matrix of matrices is obtained, it means, from each position of the area, the procedure throws one co-occurrence matrix. Taking the example of figure 4.2.1 as a reference, a total of 8 matrices is obtained, each one independent from the others. Figure 4.2.2 shows a graphic representation of algorithm 4.2.1.

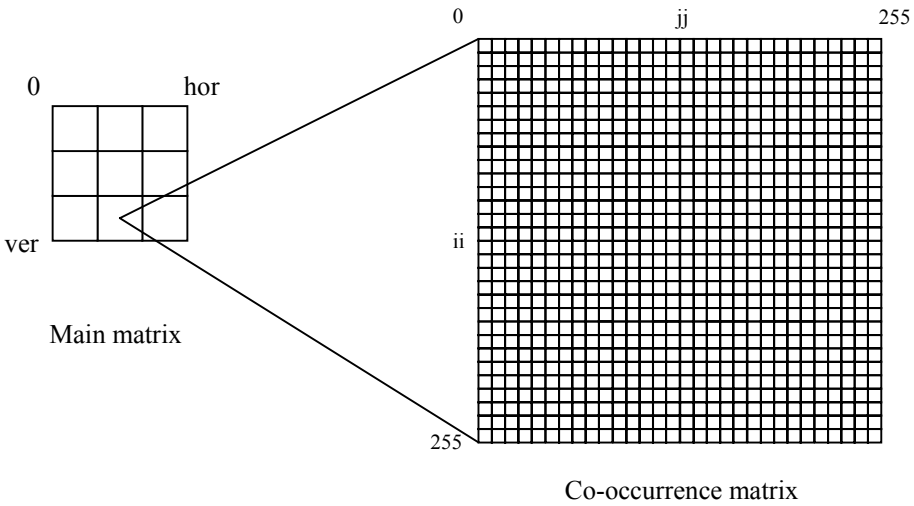


Figure 4.2.2 shows an example of the matrix of matrices, each cell of the main matrix, except the middle, contains a co-occurrence matrix (256 x 256).

The coordinates of the pivot pixel are “pi”, “pj” and “pi+disti”, “pj+distj” correspond to the position of the neighbour pixel. The distance between them varies to cover the whole working area, hence the values of “disti” and “distj”. The inner loops stops by reaching the last possible neighbour pixel of the image. The procedure stops when all the matrices have been calculated for each position of the working area. Each cell of the co-occurrence matrix contains the number of times the shade “ii” is related to the shade “jj”, namely co-occurs.

Once the co-occurrence matrices are completed, it is necessary to normalise them, since the probabilities will work between 0 and 1. A sum of all the values is made and the matrix is divided by the total sum. Each matrix is normalised separately due they are independent from each other.

$$NCM = \frac{CM}{\sum_{i=0}^{255} \sum_{j=0}^{255} v_{(i,j)}}$$

“NCM” is the normalised co-occurrence matrix, “CM” is the original matrix and “v” represents each cell value of the matrix with indices “i” and “j”.

It is necessary to make the co-occurrence matrices cumulative and then normalise them to acquire distribution functions. The process is as follows: there are two loops from i=0 to 255 and from j=1 to 255 that adds the values of the matrices at the actual position (j) with the prior position (j-1). The next step is to normalise the arrays, the process divides the whole matrices by the value in the last position (255).

4.2.1.2 Image Reconstruction

Until here, the processes only acquire the statistics of the image; the second part of the experiment handles the construction of a new image based on the obtained information that lies on the co-occurrence matrices. This process reconstructs the image by taking the information of the nearest pixels (working area) and comparing the values with the obtained matrices.

The procedure does not start from an empty image but it takes a margin of the original picture, which will be the reference, and then from this margin the new image is built. Figure 4.2.3 displays the *modus operandi* of the reconstruction.

Each new generated pixel gathers information from the neighbours, defined by the working area, compares the acquired values to the ones stored in the co-occurrence matrices and takes the most similar shade to fill the new image. This sequence is repeated until the last pixel of the image is covered.

The first step creates three arrays, one for each colour, which contain the compressed information of the textures already acquired from the co-occurrence matrices. These arrays, or vectors, are 256 elements long and represent the possible values that each colour channel can give to rebuild the image. Algorithm 4.2.2 shows the method that calculates these vectors.

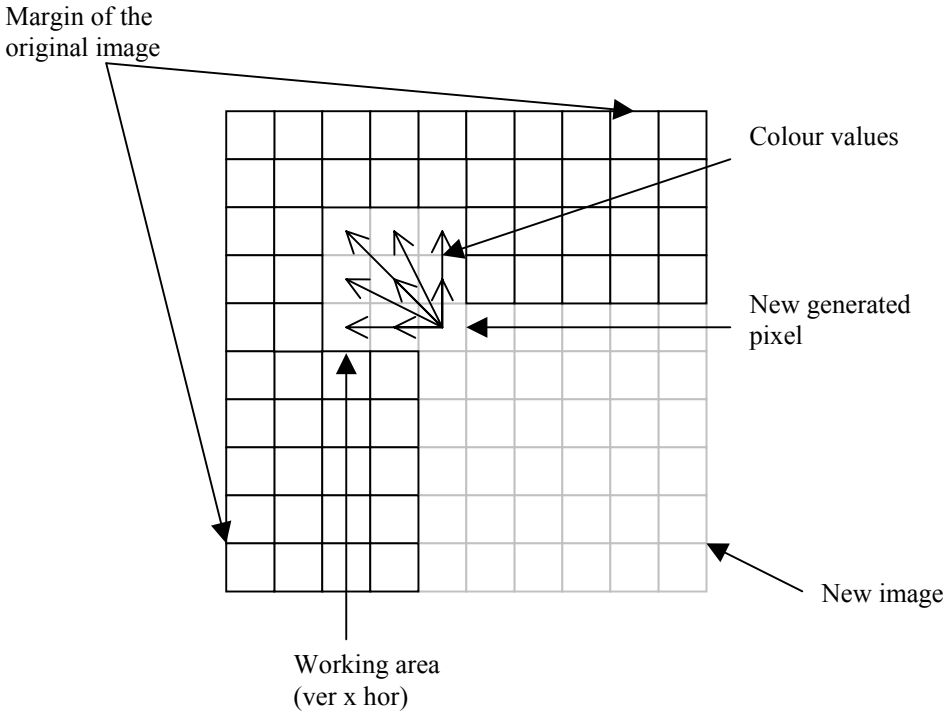


Figure 4.2.3 Reconstruction procedure.

Algorithm 4.2.2 Creation of distribution functions.

```
float Vred[] = new float[256];
float Vgreen[] = new float[256];
float Vblue[] = new float[256];
For k=0 to 255
  Vred[k]=1;Vgreen[k]=1;Vblue[k]=1;
For i=0 to ver
 For j=0 to hor
   disti=ver-1-i; distj=hor-1-j;
   For k=0 to 255
     Vred[k]*=smred[i][j][red of newImage (p-disti, q-distj)][k];
     Vgreen[k]*=smgreen[i][j][green of newImage (p-disti, q-distj)][k];
     Vblue[k]*=smblue[i][j][blue of newImage (p-disti, q-distj)][k];
```

Before the process starts, the arrays are initialized with ones. This is important since the method used to combine the different matrices is the multiplication, therefore is necessary to avoid zeros, because these could cause errors when the image is reconstructed.

The main loop is specified by the working area “ver” x “hor” and the inner loop is in charge of filling the vectors. The variables “p” and “q” represent the position of the pixel in the new image; “disti” and “distj” correspond to the distance between the pivot pixel and the neighbour pixels.

The process takes the red, green and blue value of the pixel located at the position (p-disti, q-distj), then compares these pixel values to the ones stored in the first co-occurrence matrix, which is located in the position (i,j) of the main matrix. This continues until it covers the 256 possible shades for each colour.

The co-occurrence values are multiplied to combine them and stored in the main arrays “Vred”, “Vgreen”, “Vblue”. The same procedure occurs with the second co-occurrence matrix and it continues until it covers all the matrices.

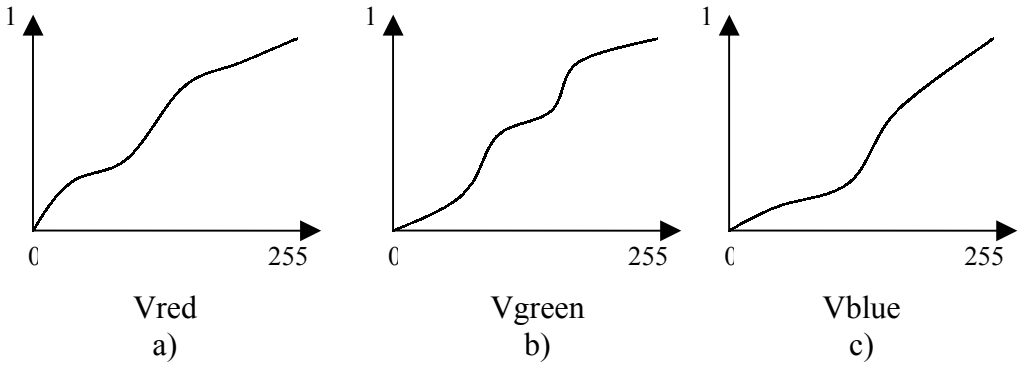


Figure 4.2.4 Examples of distribution curves. a)Red channel, b) green channel, c) blue channel.

The second part of this method makes the vectors cumulative and then normalise them. The final vectors are represented in figure 4.2.4.

The resulting vectors have the most important values for each colour channel. The process is similar to an intersection, speaking of set theory, because the insignificant values are not considered. The process creates a set of distribution functions (red, green and blue) for each pixel of the image to reconstruct. This is a recursive method and it runs until the image is completely covered. The next method is the one that chooses the colour shades to generate the image (see algorithm 4.2.3).

Algorithm 4.2.3 Colour selection.

```
multip(p,q);
triple[red]=0;triple[green]=0;triple[blue]=0;
r =Math.random();flag=true;
For i=0 to 255 and flag =true
 if(Vred[i]>=r)
  triple[red]=i;flag=false;
r =Math.random();flag =true;
For i=0 to 255 and flag =true
 if(Vgreen[i]>=r)
  triple[green]=i; flag=false;
r =Math.random();flag =true;
For i=0 to 255 and flag =true
 if(Vblue[i]>=r)
  triple[blue]=i; flag=false;
```

At the beginning the procedure calls the multiplication method “multip”, the one that calculates the colour vectors “Vred”, “Vgreen” and “Vblue”. Once the vectors are obtained for the position (p,q), a random number is generated and compared with the values of the vector “Vred”. When the value of “Vred” is bigger than the random number, the loop stops (flag=false) and the index of “Vred” (i) is stored in the array “triple” in the first position. The index “i” represents the value of the red shade.

The process is repeated for “Vgreen” and “Vblue” and their indices are saved in the second and third position of the array “triple”. The last procedure is called “fill”, and is in charge of creating the image. Its code is shown in algorithm 4.2.4.

Algorithm 4.2.4 Filling the image.

```
For x=horiz to imagewidth-margin
 For y=verti to imageheight-margin
 {
    construct(x,y);
    newimg[x][y]= triple[red] + triple[green] + triple[blue];
 }
```

This loop leaves a margin of the original image and fills the rest using the information obtained in previous procedures. The margin width is given by the values “horiz” and “verti”. It calls the construct procedure (algorithms 4.2.2 and 4.2.3), and obtains the shade values for each channel.

It remains only to mix the red, green and blue shades to generate the desired colour and fill the pixel at the position (x,y). This is the last procedure and as a result gives the new generated image.

4.2.1.3 Results

Following, the co-occurrence matrices method is used to learn different textures and later reconstruct them. A wide variety of images is analysed, from plain textures to watermarks, wooden patterns and bricks, obtaining significant results from the experiments.

4.2.1.3.1 First Category: Plain Textures

At the beginning plain textures are analysed. Figure 4.2.5 shows an example of plain texture, a grey sample created with Microsoft Paint 5.1 ™.

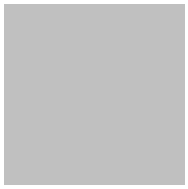


Figure 4.2.5 Monochromatic sample.

The parameters for the experiment are, margin=10 pixels, which remains constant along the regular textures, and the working area that varies in different procedures. For this experiment the working area has a width of 2 pixels and height of 2 pixels too. This is a basic texture; it is just one plain colour, so the size of the working area does not affect the results. The red, green and blue components of this image share the same value, 192, where 0 is black and 255 is white. The cell (192,192) has the totality of the pixels, which is, for this image, 32004; the rest of the matrix cells are empty. After acquiring information (co-occurrence matrices), an area within the image is erased.

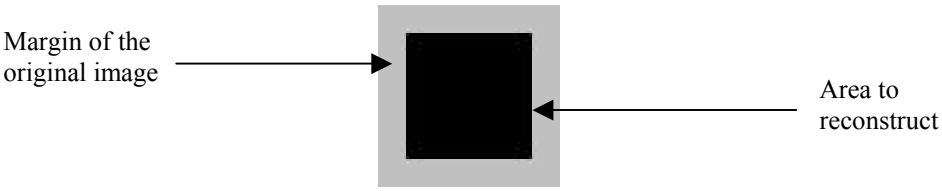


Figure 4.2.6 Intermediate step of the procedure.

The procedure takes a random number between 0 and 1 and compares it with the distribution functions (red, green and blue), which are histogram-like curves; these curves are cumulative and normalised, so their range lies also between 0 and 1. When the value of the function is greater than the random number, that curve position is taken, which is actually, the colour value.

For this image the distribution curves are exactly the same, since it is a monochromatic image. It occurs the same as in the co-occurrence matrix; all the values are in the position 192, so its distribution curves look similar to a step function, see figure 4.2.7.

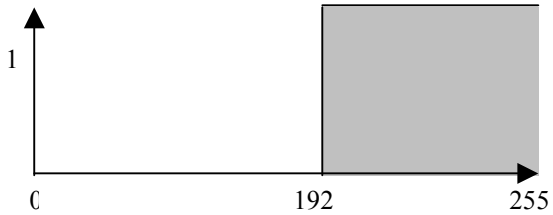


Figure 4.2.7 Distribution function (the same curve for red, green and blue channels).

Since there are no other possibilities as the value 192 for the three colours, the image is easy to restore to its original state, so at the end of the reconstruction an identical copy of the original is achieved:

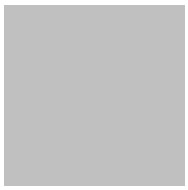


Figure 4.2.8 Reconstruction of the monochromatic sample.

4.2.1.3.2 Second Category: Regular Textures

The next image to be treated is a vertical pattern (created with Microsoft Paint 5.1 ™), which is also well organized but not as simple as the grey picture.

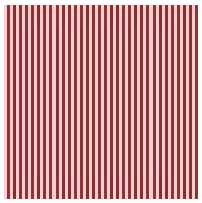


Figure 4.2.9 Vertical pattern.

The working area for this test has height=2 pixels and width=2 pixels. Like in the latter example, the size of the working area still does not play an important role, because the texture remains regular. The green and blue co-occurrence matrices are very similar due the image is red dominant. The red co-occurrence matrix has a shorter range of values. It follows the same procedure and erases the inner part of the image, once it is acquired the information for the co-occurrence matrices.

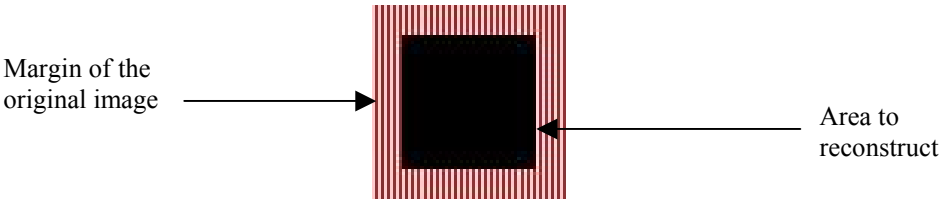


Figure 4.2.10 Intermediate step of the procedure.

The same procedure is effectuated; a random number between 0 and 1 is taken and compared with the distribution functions (red, green and blue). When the value of the function is greater than the random number, that index is taken as colour value.

The distribution functions for this image are certainly wider than the monochromatic texture. However, they remain sharp enough to give the precise values to reconstruct the image. The colour histograms for this image, see figure 4.2.11., give an approximate idea of the distribution functions.

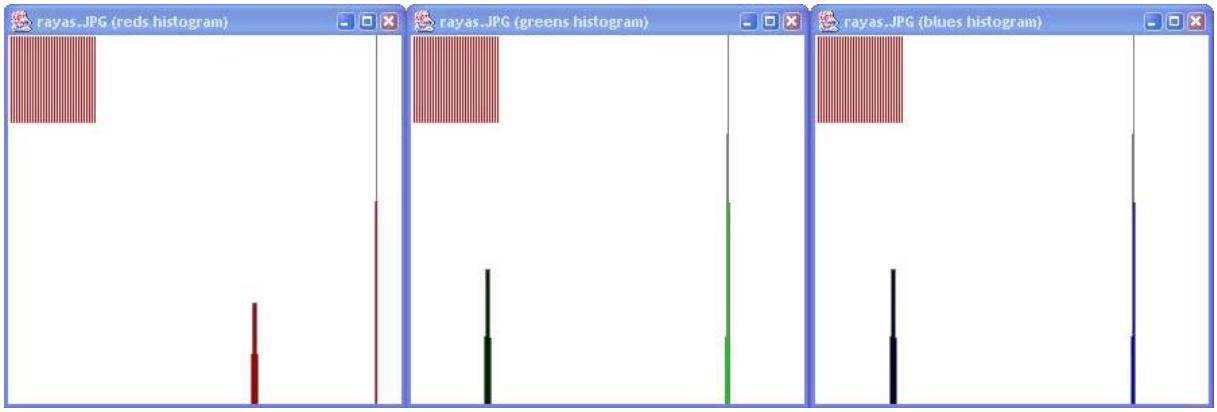


Figure 4.2.11 Histograms for the vertical pattern. a)red histogram, b) green histogram and c) blue histogram.

As observed, the colour values are well ordered in just a small range of the scale; that makes the image easy to reconstruct, due the sharpness of the distribution functions. The values tend to the "bright" side of the scale because of the white stripes; however, green and blue are also located in the dark side (37 to 40 in the 255 scale) while red gets a higher value (166 to 169), which indicates that the image is red dominant. After the restoring process a perfectly reconstructed image is obtained.

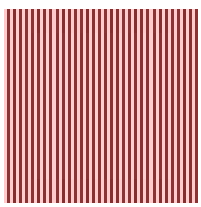


Figure 4.2.12 Vertical sample reconstructed with co-occurrence matrices.

4.2.1.3.3 Third Category: Patterned Textures

Following is another type of texture, this time is not so well ordered and there are also more colour shades, which made it more complicated to restore, for example, the wooden pattern shown below. This image is not homogeneous, like the last two textures. It is horizontal oriented, but not regular.

Figure 4.2.13 "Pinelumb" sample (obtained from MicrosoftOffice Textures Samples).

The first step calculates the co-occurrence matrices, after that, the image centre is erased and later reconstructed, applying the restoring method. This "pinelumb" texture has wider distribution functions than the previous pictures. The working area has width=2 pixels and height=2 pixels. Histograms give an approximation to the distribution functions, and as observed in figure 4.2.14, the image colours are quite good aligned, that is, the range of values is small enough to make a smooth restoration of the image.

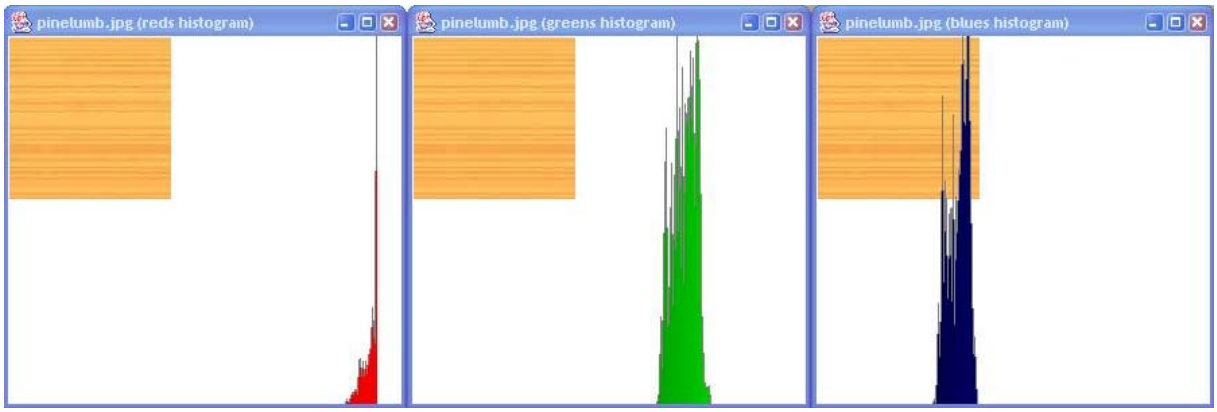


Figure 4.2.14 Histograms for the "pinelumb" pattern. a)red channel, b) green channel and c) blue channel.

The majority of the red values lie on the bright edge, this is 255, which makes the red distribution function narrow and easier to determine. Green and blue values lie in a wider range but still in a compact set. After applying the restoring method, the reconstructed image is obtained, see figure 4.2.15.

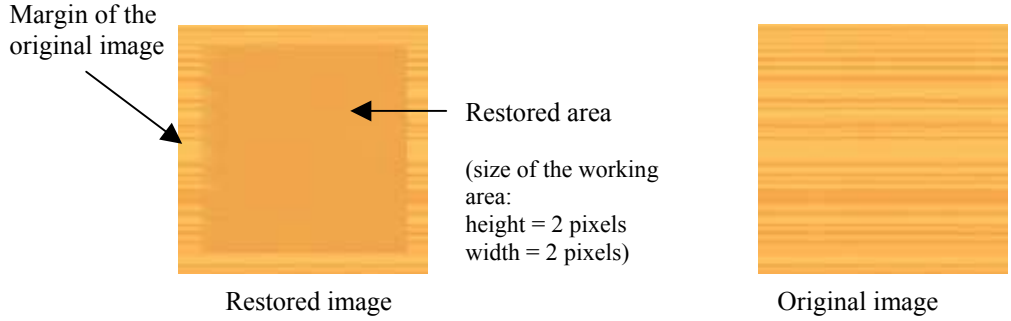


Figure 4.2.15 Comparison between obtained image(left) and the original sample(right).

A valuable hint to improve the restoring process is the lack of stripes, which means that the problem may lie in the working area, since this is too narrow to detect the stripes. For the next calculation, the working area has a height and width of three pixels. The co-occurrence matrices are increased in number, as well as the grasp of their neighbourhood relations. The results are displayed in the next figure.

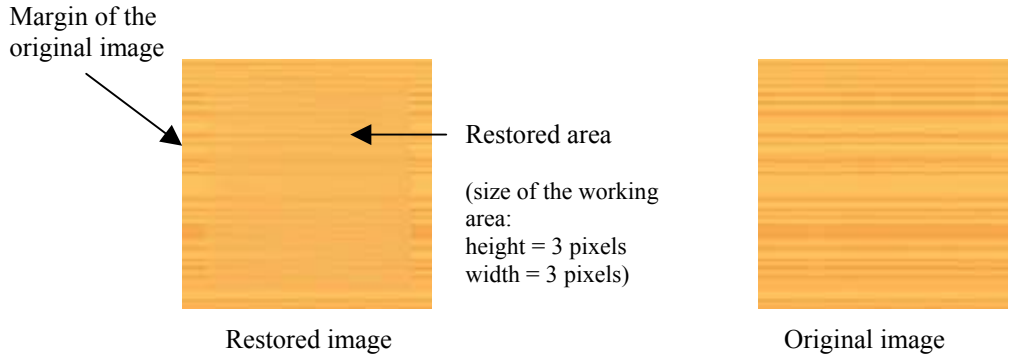


Figure 4.2.16 Comparison between restored image(left) and the original sample(right).

An improvement in the final image is clearly noticeable. The stripes are present on this image and the colour is far more similar to the original than the previous experiment. For the last text, even a bigger working area is defined in order to improve the results. Width and height of the working window have now 4 pixels.

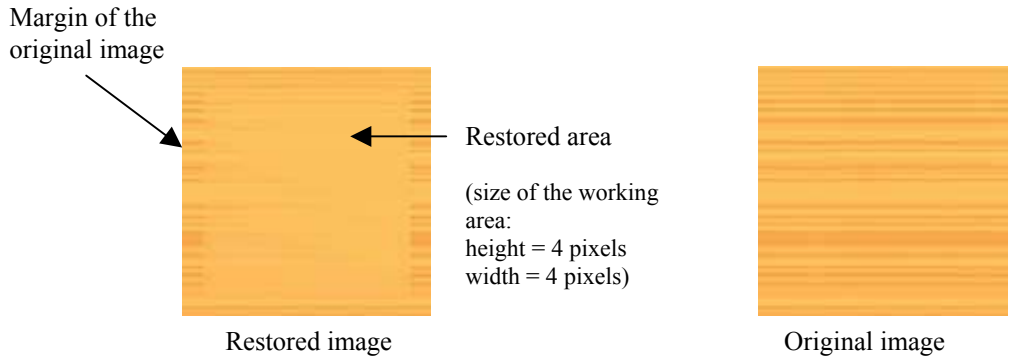


Figure 4.2.17 Comparison between restored image (left) and the original sample(right).

The quality of the final image decreases instead of improving. This means that there is no use in increasing the size of the working area, but to look for a different approach to enhance the accuracy; this will be explained in the second stage of the analysis (chap. 4.2.2).

The next image to analyse is a small sample from Neptune surface, see figure 4.2.18.

Figure 4.2.18 Neptune sample taken from "Planetary Textures for model construction" website [44].

As observed in the sample, there is a small irregularity on the bottom right corner, which could be eliminated and restored with the original pattern of the image. The initial parameters for this sample are height and width of 2 pixels.

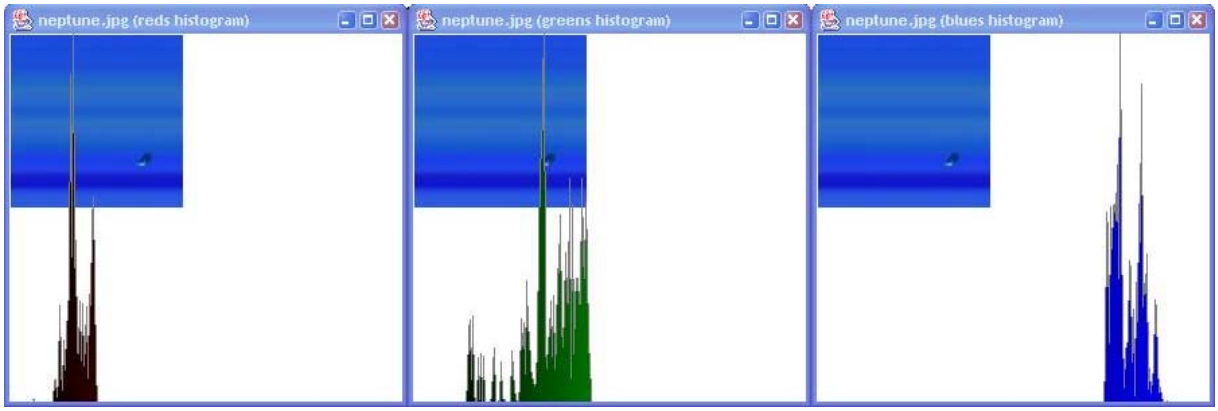


Figure 4.2.19 Histograms for Neptune sample. a)red histogram, b) green histogram and c) blue histogram.

Histograms in figure 4.2.19 show that this image is more complex than the previous three samples, its colour range is wider and its colours are not concentrated in just one peak or set, but in various sections in the spectrum, which makes the sample more difficult to restore. The result for this reconstruction method is shown in figure 4.2.20.

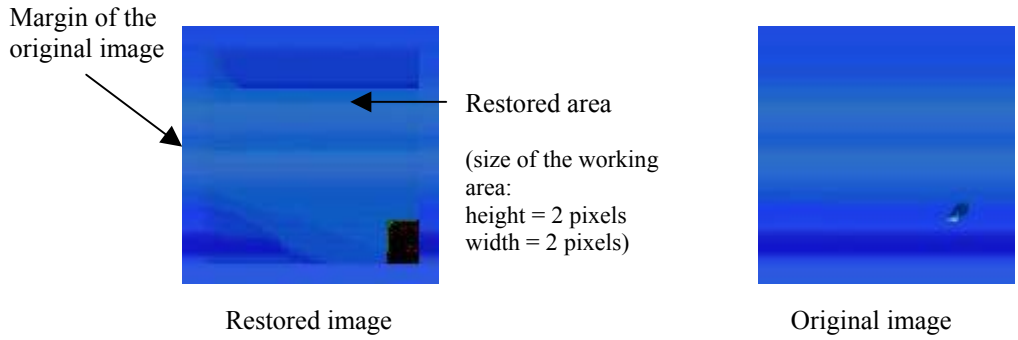


Figure 4.2.20 Comparison between the obtained image(left) and the original sample(right).

The resulting image is not good enough, the small irregularity has become bigger and the colours are blurred, although they are similar to the original image. This leads to an increasing in the size of the working area to improve the distribution of the stripes along the picture. After considering the latter result, a new test is proved but this time with a bigger working area, 3x3 pixels. Figure 4.2.21 shows the result of the analysis with a height and width of three pixels for the working window.

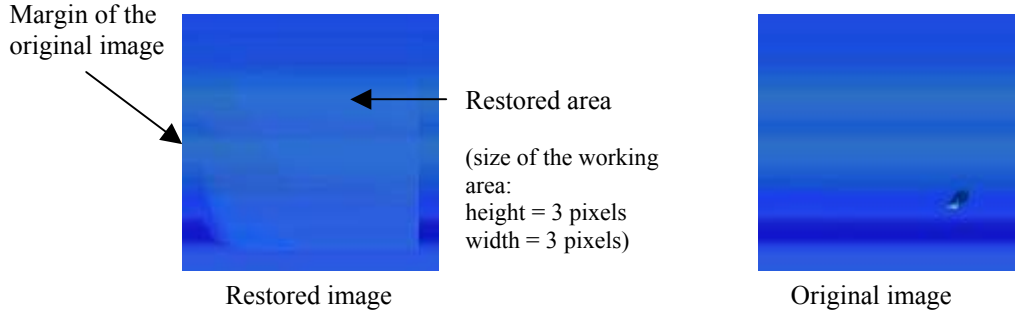


Figure 4.2.21 Comparison between the obtained image(left) and the original sample(right).

An improvement in the image is acquired thanks to the larger working area. The upper part of the image is almost an exact copy of the original; however, the bottom still needs to be refined, although the irregularity is not present anymore in the final picture.

A final test with this sample with a bigger working area of 4x4 pixels is shown in figure 4.2.22.

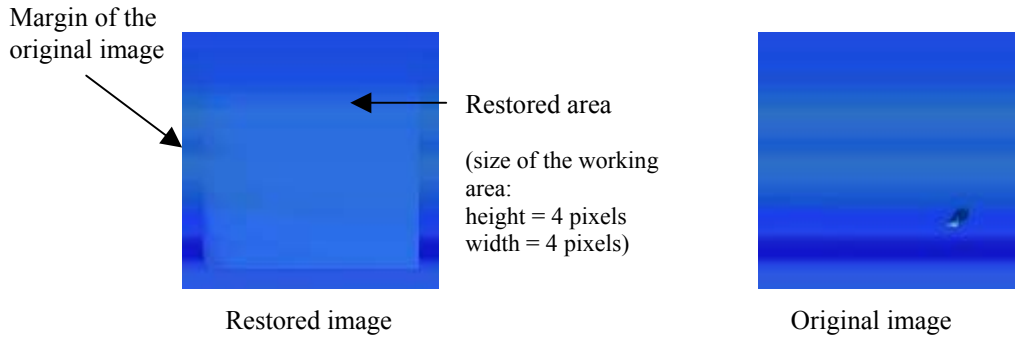


Figure 4.2.22 Comparison between the obtained image(left) and the original sample(right).

As expected, the result decreases in quality due to the large amount of information. When calculating 4x4 matrices, some of the stored values are redundant, this is inexpedient.

The next texture to analyse is a bricks sample, see figure 4.2.23.

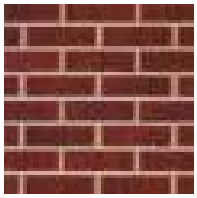


Figure 4.2.23 Bricks sample.

Its colours are not so well organized as in the previous images, as observed in its histograms, see figure 4.2.24. The range of values is wide for each colour. Red channel covers a wider area than greens and blues, and is the dominant colour.

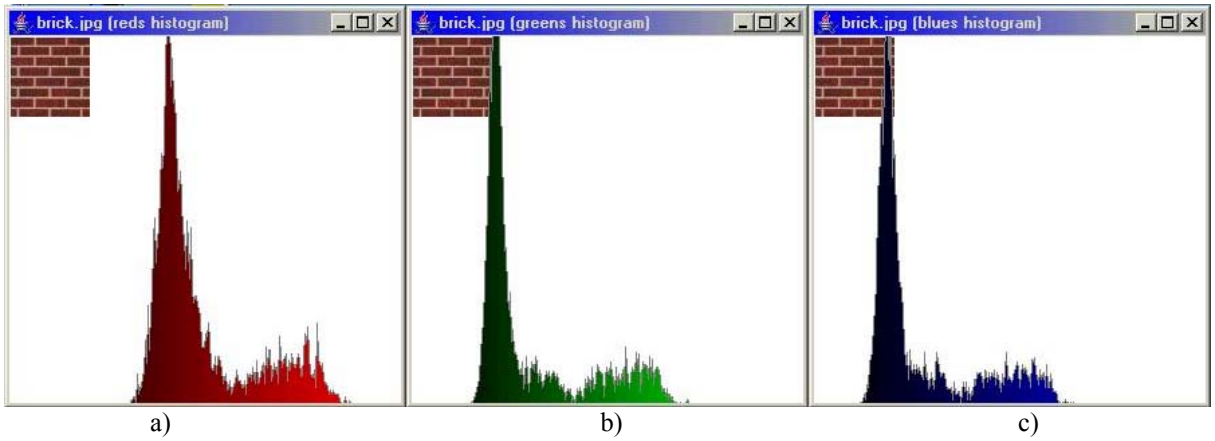


Figure 4.2.24 Histograms for the bricks sample. a)red histogram, b) green histogram and c) blue histogram.

The working area for this sample is 2x2 pixels wide. Figure 4.2.25 shows the results obtained.

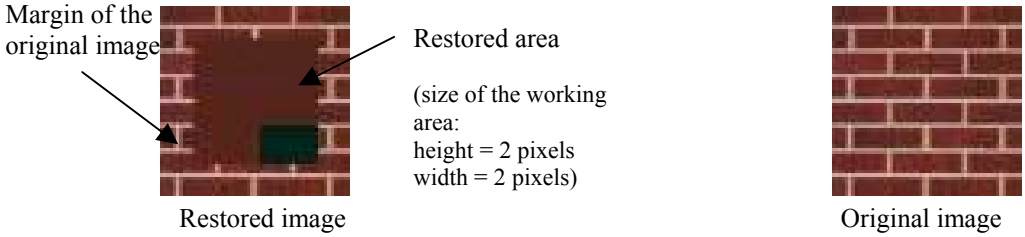


Figure 4.2.25 Brick sample restored(left) and original sample(right).

As observed, the results are not satisfactory. It fills the image with a solid colour, that is similar to the most dominant shade in the original sample, but this is not enough. A small irregularity can also be observed in the lower right section.

For the next test, the working area is increased to scan more information to the co-occurrence matrices. Figure 4.2.26 displays the results obtained with a working area of 3x3 pixels.

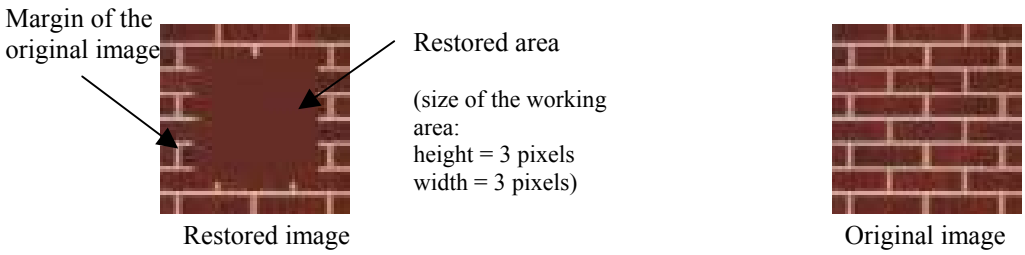


Figure 4.2.26 Results obtained with a 3x3 pixel working area(left) and original sample(right).

The quality of the image increases, its colour is almost the same as the dominant in bricks but the image is still far from the original sample. The last try for this sample is with a working area of 4x4 pixels. The results are displayed in figure 4.2.27.

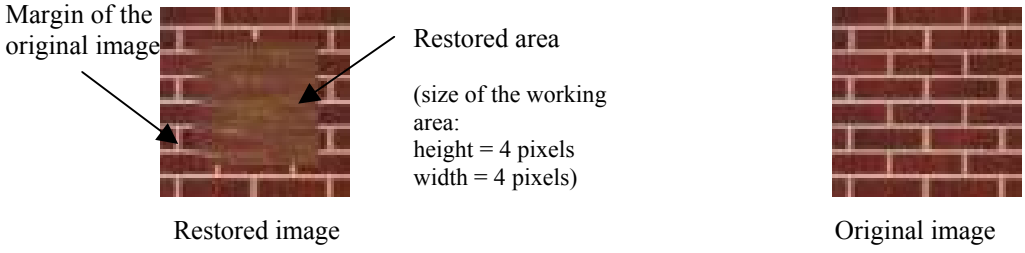


Figure 4.2.27 Restored brick sample (left) and original sample (right).

These results are better than the previous two. It is not a solid colour anymore, it has different shades and also the traces of the brick are slight recognized.

For this texture, the bigger the working area, the better the results. In the previous samples (Neptune and the “pinelumb” texture), a working area of 4x4 is used and due to the amount of information, the quality of the results decreases.

4.2.1.3.4 Fourth Category: Irregular Patterned Textures

This category contains images with an established pattern, but the colours are not so well organized like in the last pictures. The next image to test is another wall sample, but the bricks are smaller and the colours are not so uniform as in the previous texture. Figure 4.2.28 shows this new sample.

Figure 4.2.28 Wall sample.

This image pattern contains some strong colour and contrast changes. There are some darker areas and the shades are not uniform like the previous sample (see figure 4.2.23). The following figure shows the histograms for this sample.

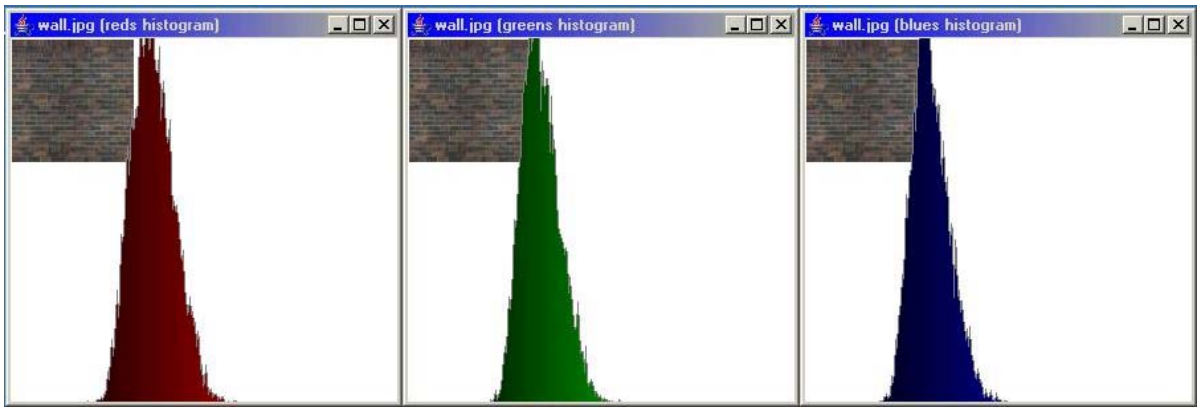


Figure 4.2.29 Histograms for the wall sample. a)red histogram, b) green histogram and c) blue histogram.

As observed in figure 4.2.29, the colour distribution is balanced; the spectra are similar and cover almost the same range of values. The first experiment for this texture has working area of 2x2 pixels and the margin is set to 25 pixels wide. Results are shown in figure 4.2.30.

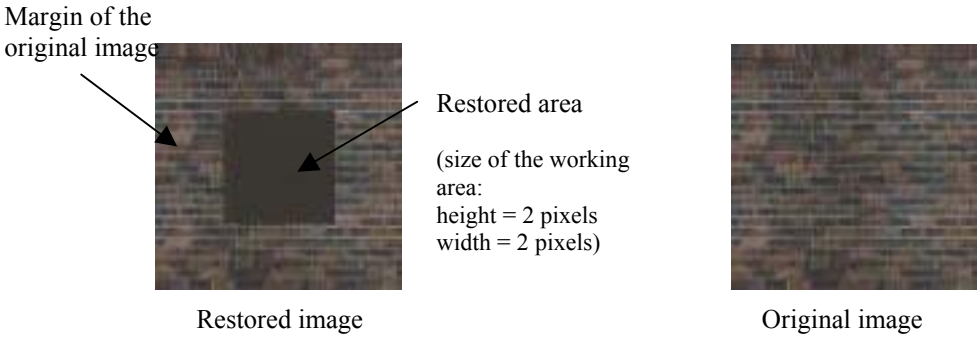


Figure 4.2.30 Reconstructed wall sample (left) and original sample (right).

The image is filled with a solid colour and the traces of other shades or patterns are not present. This occurs due to the complexity of the image, higher than the previous samples. The amount of data is not enough to achieve a better reconstruction. For the next test, a bigger working area, 3x3 pixels is applied. The margin of 25 pixels wide remains constant for the following experiments with this texture. Figure 4.2.31 shows the obtained results.

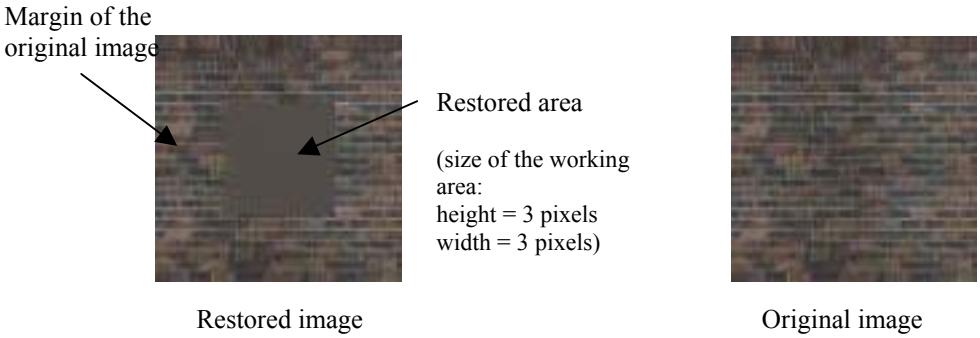


Figure 4.2.31 Obtained image (left) and original sample (right).

As observed, the quality of the reconstruction increases. It is still one solid colour filling, but this is nearer to the dominant shades of the picture than the obtained in the previous testing (see figure 4.2.30). The next test is calculated with a working area of 4x4 pixels. Figure 4.2.32 shows the obtained results.

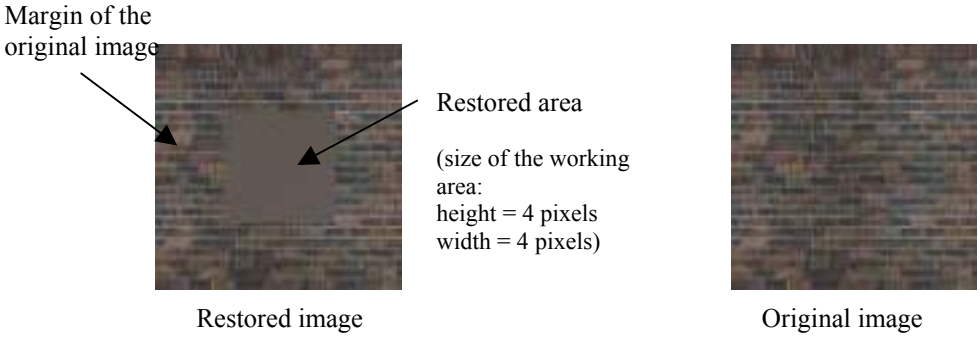


Figure 4.2.32 Reconstructed sample (left) and original sample (right).

There is also a one colour filling present, but there is also a slight trace of pattern in the upper right section of the image. This leads to the conclusion of expanding the working area, not only in size, but in range. A better description is given in section 4.2.2 of this chapter.

The next image to analyse is a sample of Jupiter, taken also from "Planetary Textures for model construction" website [44], see figure 4.2.33.

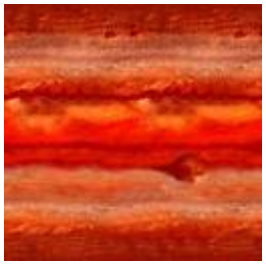


Figure 4.2.33 Jupiter sample taken from "Planetary Textures for model construction" website.

The working area for this test is 2 pixels wide and 2 pixels high. It is a more complex image than the Neptune sample due to the wider variety of colour shades. Figure 4.2.34 shows the histograms for this sample.

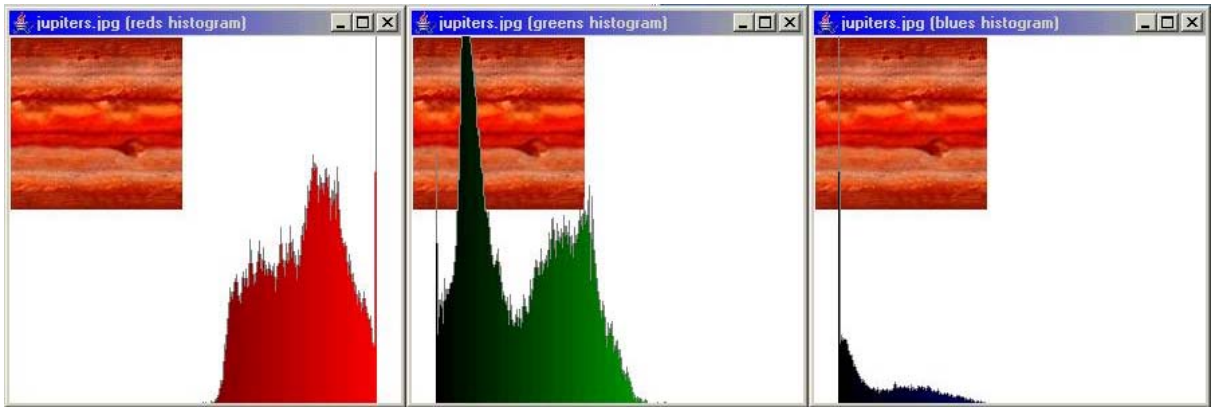


Figure 4.2.34 Histograms for Jupiter sample. a)red histogram, b) green histogram and c) blue histogram.

This sample histograms are nothing but simple. The red range is wide and uniform, the green channel is also wide but irregular and the blue channel is almost non existent. Figure 4.2.35 shows the result for this experiment with the 2x2 pixels working window and margin of 35 pixels wide, which remains constant in the following experiments.

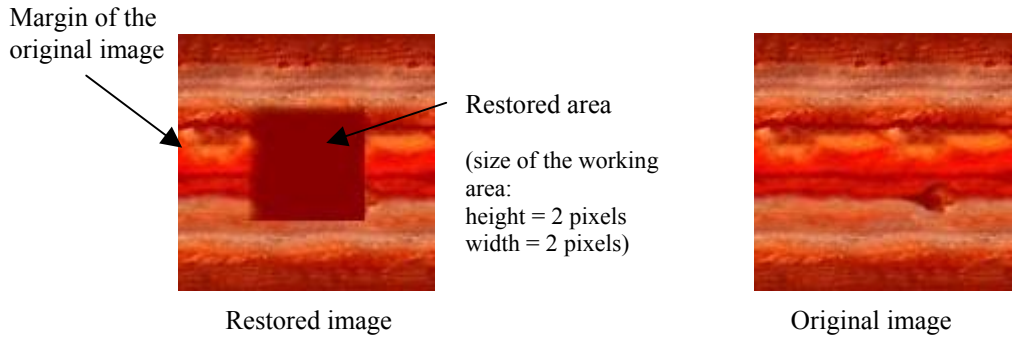


Figure 4.2.35 Comparison between the reconstructed image(left) and the original sample(right).

As expected, the reconstruction for an irregular sample like Jupiter is not so good as the results obtained with regular textures. However, it is clear that the higher the complexity of the image, the wider the working area has to be to collect important information to the restoring process. Next figure shows the results for this image with a wider area of 3x3 pixels.

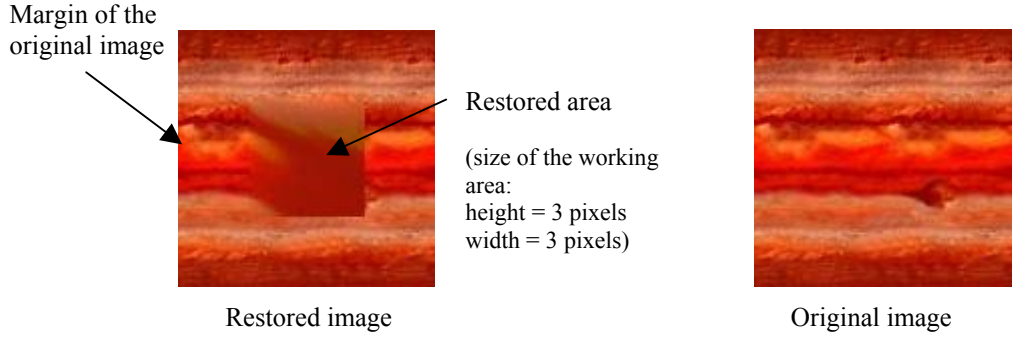


Figure 4.2.36 Reconstruction of Jupiter sample(left) and the original sample(right).

There are slight traces of the pattern located on the left side, but they disappear and do not cover the entire image. The rest of the image is filled with a solid colour.

The last test for this sample is calculated with a 4x4 pixels working area. Its results are displayed in figure 4.2.37.

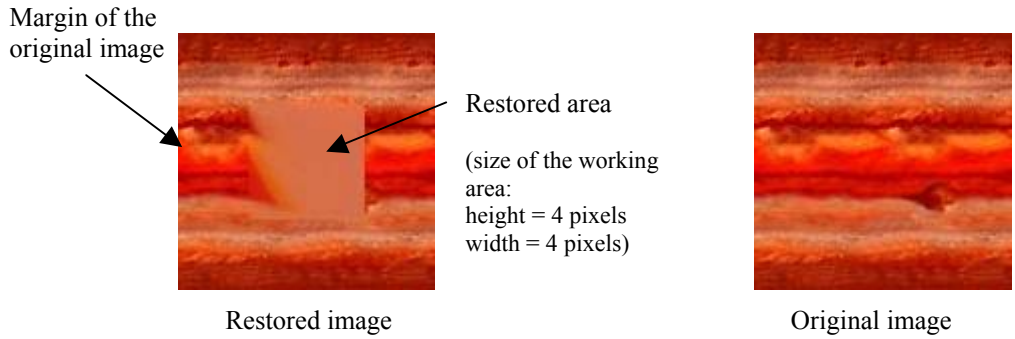


Figure 4.2.37 Restored sample(left) and the original sample(right).

This results show also traces of patterns, but they disappear after a short distance from the start point (upper-left). In the following section (4.2.2) a different working area is used. This new area covers a wider range of pixels in all directions, so the co-occurrence matrices obtain more data related to the observed pixel.

4.2.2 Second approach: Wide Working Area of Co-occurrence Matrices

4.2.2.1 Acquiring information

After evaluating the results, is clear that the working area is a very important parameter for this experiment, however it only collects information of the lower right quadrant, so next idea is to increase the size of the working area and include all the directions surrounding the pivot pixel; this way more relevant information can be obtained and stored in the co-occurrence matrix during the learning process. The modifications made to the matrices are shown below.

```
float smred [ver*2-1][hor*2-1][256][256];
float smgreen [ver*2-1][hor*2-1] [256][256];
float smblue [ver*2-1][hor*2-1] [256][256];
```

Where *ver*2-1* and *hor*2-1* correspond to the size of the working area and the next parameters represent the range of values for each colour channel, this varies from 0 to 255.

As explained in the first procedure, co-occurrence values of the image are taken from the three colour channels; these values are obtained as the working area scans the image. The direction of this co-occurrence is not only the eight nearest pixels, but the whole working area with the adjacent pixels, as shown in figure 4.2.38. The working window scans the image, extracts the colour components and stores them in the co-occurrence matrices.

Algorithm 4.2.1 shows the obtaining of co-occurrence matrix but only for the lower right pixels sector. It is necessary to add the other three quadrants to calculate the new working area. This new code is shown in algorithm 4.2.5, which is the modified version of algorithm 4.2.1.

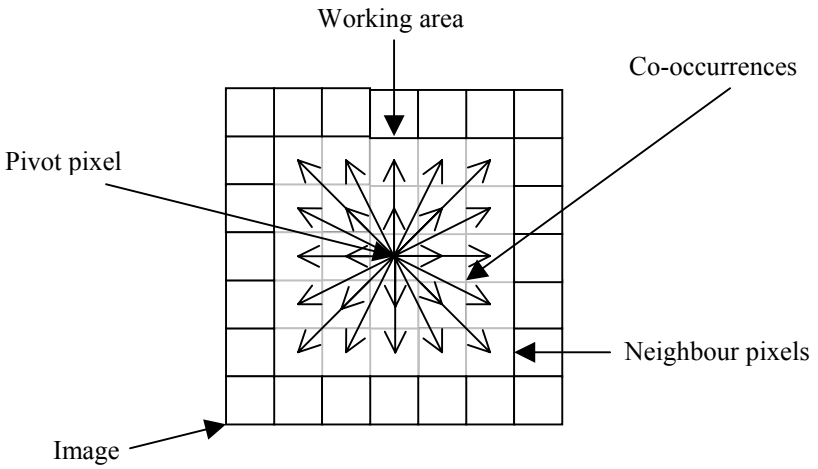


Figure 4.2.38 New working area for the second approach.

Algorithm 4.2.5 Obtaining of co-occurrence matrices with wide working area.

```
For i=0 to ver
 For j=0 to hor {
If (i<ver-1) or (j < hor-1) {
  disti=ver-1-i; distj=hor-1-j;
   /*************** first sector calculation*************/
   For pi=0 to Imageheight - disti
    For pj=0 to Imagewidth - distj  {
     smred[i][j][red of Image(pi, pj)] [red of Image(pi+disti, pj+distj)] +=1;
     smgreen[i][j][green of Image(pi, pj)] [green of Image(pi+disti, pj+distj)] +=1;
     smblue[i][j] [blue of Image(pi, pj)] [blue of Image(pi+disti, pj+distj)] +=1;
   }
/*************** second sector calculation*************/
   For pi=0 to Imageheight - disti
    For pj=Imagewidth downto distj   {
     smred[i][2*hor-j][red of Image(pi, pj)] [red of Image(pi+disti, pj-distj)] +=1;
     smgreen[i][2*hor-j][green of Image(pi, pj)] [green of Image(pi+disti, pj-distj)] +=1;
     smblue[i][2*hor-j] [blue of Image(pi, pj)] [blue of Image(pi+disti, pj-distj)] +=1;
   }
/*************** third sector calculation*************/
   For pi=Imageheight downto disti
    For pj=0 to Imagewidth - distj   {
     smred[2*ver-i][j][red of Image(pi, pj)] [red of Image(pi-disti, pj+distj)] +=1;
     smgreen[2*ver-i][j][green of Image(pi, pj)] [green of Image(pi-disti, pj+distj)] +=1;
     smblue[2*ver-i][j] [blue of Image(pi, pj)] [blue of Image(pi-disti, pj+distj)] +=1;
   }
/*************** fourth sector calculation*************/
   For pi=Imageheight downto disti
    For pj=Imagewidth downto distj   {
     smred[2*ver-i][ 2*hor-j][red of Image(pi, pj)] [red of Image(pi-disti, pj-distj)] +=1;
     smgreen[2*ver-i][2*hor-j][green of Image(pi,pj)] [green of Image(pi-disti,pj-distj)]+=1;
     smblue[2*ver-i][ 2*hor-j] [blue of Image(pi, pj)] [blue of Image(pi-disti, pj-distj)] +=1;
   }
 }
}
```

The working area covers the image and gets the number of occurrences. The main loops are defined by the variables “ver” and “hor”. The scanning process is divided in four sectors, lower-right, lower-left, upper-right and upper-left, and the resulting matrices are stored to their correspondent location.

As explained in the previous experiment, a matrix of matrices is obtained, but this time is a bigger matrix; for the example in figure 4.2.38 which has height and width of five pixels, 24 matrices are obtained (the pivot pixel is not included in the calculations).

The procedure works exactly the same as the one of the previous experiment; the only difference is that now calculates four quadrants instead of only one. Figure 4.2.39 gives an example of the obtained matrices:

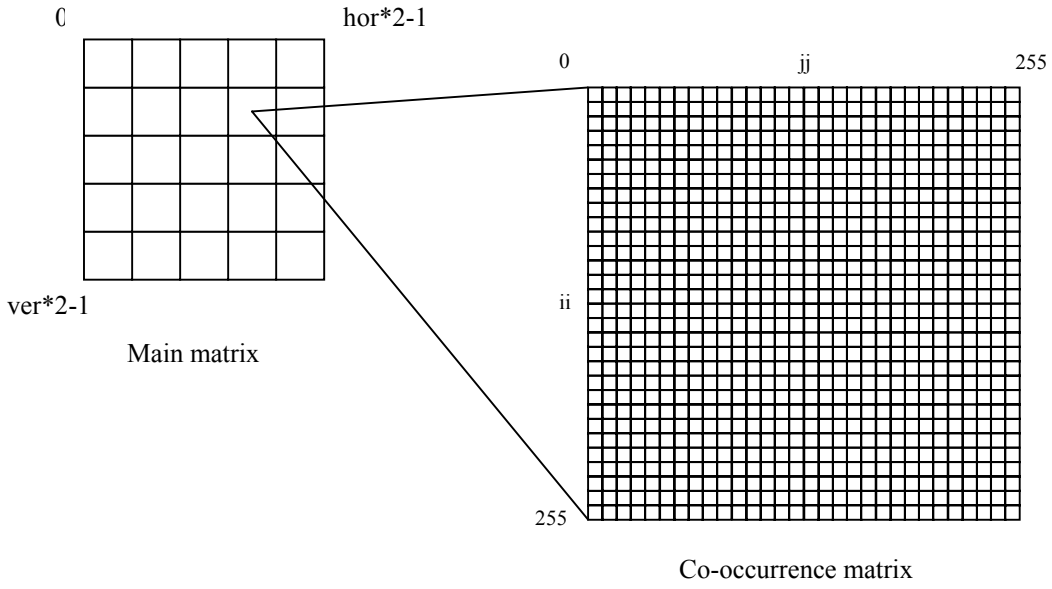


Figure 4.2.39 shows an example of the matrix of matrices, each cell of the main matrix, except the middle, contains a co-occurrence matrix (256 x 256).

Each cell of the co-occurrence matrix contains the number of times the shade “ii” is related to the shade “jj”. After the matrices are stored and filled with information, it is necessary to normalise them because the probabilities that will be used lie between 0 and 1. Due the independence of the matrices, each of them is normalised individually.

$$NCM = \frac{CM}{\sum_{i=0}^{255}\sum_{j=0}^{255} v_{(i,j)}}$$

“NCM” is the normalised co-occurrence matrix, “CM” is the original matrix and “v” represents each cell value of the matrix with indices “i” and “j”.

Another necessary process consists in making the co-occurrence matrices cumulative and then normalise them again to acquire distribution functions.

4.2.2.2 Image Reconstruction

Until here, only the process of acquiring information was described, the next part of the experiment rebuilds the image; its *modus operandi* takes the information of the working area and compare it with the one stored in the co-occurrence matrices. However it is slightly different than the previous experiment (see chap.4.2.1) due to the working area, which is wider than the one used in the first experiment. As it was described in the first approach, the procedure takes the margin of the original picture as a reference, and then reconstructs the image from that information. Figure 4.2.40 shows how the reconstruction method works.

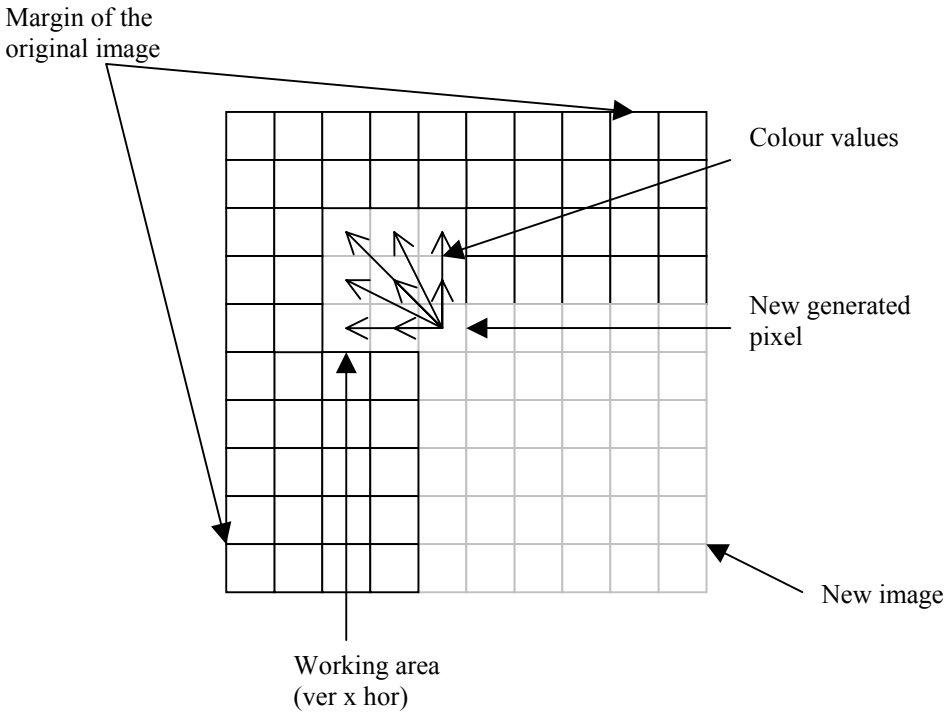


Figure 4.2.40 Reconstruction procedure.

Every new generated pixel obtains information from its neighbours, which are defined by the working area, and compares their content to the values in the co-occurrences matrices. This process is repeated until it reaches the height and width limits of the image. A more detailed description is given below.

It is necessary to create three colour arrays (red, green and blue), which contain the information stored in the co-occurrence matrices. Algorithm 4.2.2 covers only one of the four sectors in which is divided the new the working area. Following is the new program that calculates the four sectors of the new working area.

Algorithm 4.2.6 Creation of distribution functions with wide working area.

```
float Vred[] = new float[256];
float Vgreen[] = new float[256];
float Vblue[] = new float[256];
For k=0 to 255
{
 Vred[k]=1;Vgreen[k]=1;Vblue[k]=1;
}
```

```
For i=0 to ver
 For j=0 to hor
   disti=ver-1-i; distj=hor-1-j;
  For k=0 to 255
 {
/**************** first sector calculation**************/
      Vred[k]*=smred[i][j][red of newImage (p-disti, q-distj)][k];
      Vgreen[k]*=smgreen[i][j][green of newImage (p-disti, q-distj)][k];
      Vblue[k]*=smblue[i][j][blue of newImage (p-disti, q-distj)][k];
/**************** second sector calculation**************/
     Vred[k]*=smred[i][ 2*hor-j][red of newImage (p-disti, q-distj)][k];
     Vgreen[k]*=smgreen[i][ 2*hor-j][green of newImage (p-disti, q-distj)][k];
     Vblue[k]*=smblue[i][ 2*hor-j][blue of newImage (p-disti, q-distj)][k];
/**************** third sector calculation**************/
     Vred[k]*=smred[2*ver-i][j][red of newImage (p-disti, q-distj)][k];
     Vgreen[k]*=smgreen[2*ver-i][j][green of newImage (p-disti, q-distj)][k];
     Vblue[k]*=smblue[2*ver-i][j][blue of newImage (p-disti, q-distj)][k];
/**************** fourth sector calculation**************/
     Vred[k]*=smred[2*ver-i][ 2*hor-j][red of newImage (p-disti, q-distj)][k];
     Vgreen[k]*=smgreen[2*ver-i][ 2*hor-j][green of newImage (p-disti, q-distj)][k];
     Vblue[k]*=smblue[2*ver-i][2*hor- j][blue of newImage (p-disti, q-distj)][k];
}
```

The operation to combine the values of the different matrices is the multiplication, that is why the arrays are initialized with ones, to avoid zero values that can produce errors in the new generated image.

Variables “ver” and “hor” (working area) determine the main loop of this procedure. The inner loop is where the arrays are filled with information. Variables “p” and “q” are the coordinates of the pixel in the new image; “disti” and “distj” represent the distance between the pivot pixel and the neighbour pixels.

The process takes the colour components of the pixel located in the position (p-disti, q-distj), compares these pixel values to the ones stored in the first co-occurrence matrix, located in the position (i,j) of the main matrix. The multiplied values of the co-occurrence matrices are stored in the arrays “Vred”, “Vgreen” and “Vblue”.

The procedure continues until it covers all the matrices. The second part of this method makes the vectors cumulative and then normalises them. An example of the final distribution functions is represented in figure 4.2.41. The resulting vectors have the most important values for each colour channel. The process creates a set of distribution functions for each image pixel to reconstruct. This is a recursive method and runs until the image is fully covered.

Algorithms 4.2.3 and 4.2.4 remain exactly the same for this new approach, since the filling and reconstruction procedures are not influenced anymore by the change of the working area when calculating the co-occurrence matrices.

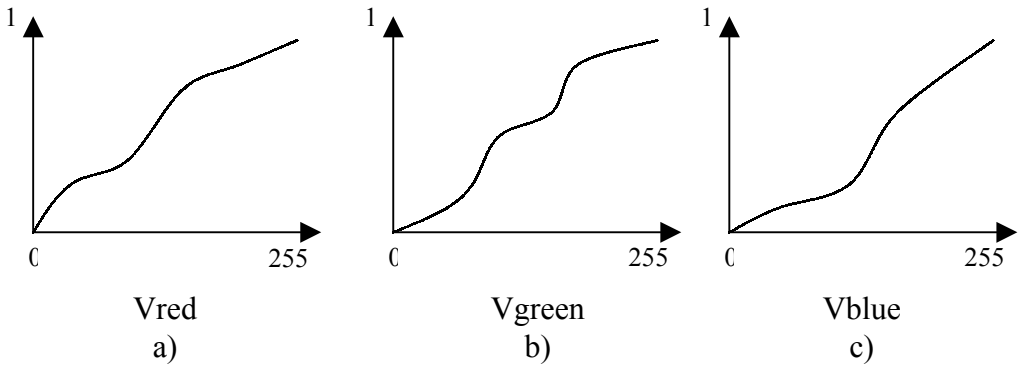


Figure 4.2.41 Examples of distribution curves. a)Red channel, b) green channel, c) blue channel.

4.2.2.3 Results

This section shows the comparison between the results obtained in the first experiment (see chap.4.2.1) to the ones in the second analysis. The explanation of the procedures is already described in the previous section, so only the results are shown in the following pages

4.2.2.3.1 First Category: Plain Textures

It starts with plain textures, like the one in figure 4.2.42 (created with Microsoft Paint 5.1 ™).

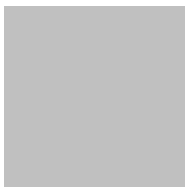


Figure 4.2.42 Monochromatic sample.

The working area for this sample has height=2 pixels and width=2 pixels. The margin for regular textures is set to 10 pixels wide. Since it is just one plain colour texture, the size of the working area does not affect the results.

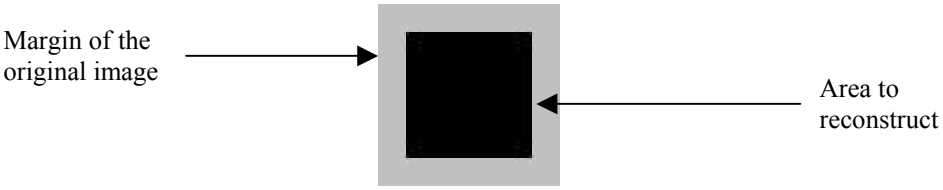


Figure 4.2.43 Intermediate step, the reconstruction takes values from the original margin.

Figure 4.2.44 Distribution function (the same curve for red, green and blue channels).

Since there are no other possibilities as the value 192 for the three colours, the image is perfectly restored to its original state, so at the end of the reconstruction an identical copy of the original is obtained.

Figure 4.2.45 Restored image with the second approach.

4.2.2.3.2 Second Category: Regular Textures

The next image to try in the experiment is a vertical pattern (created with Microsoft Paint 5.1 ™), which is also well organized but not as simple as the grey picture.

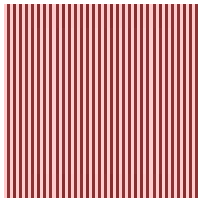


Figure 4.2.46 Vertical pattern.

The working area for this experiment has a height of 2 pixels and width of 2 pixels.

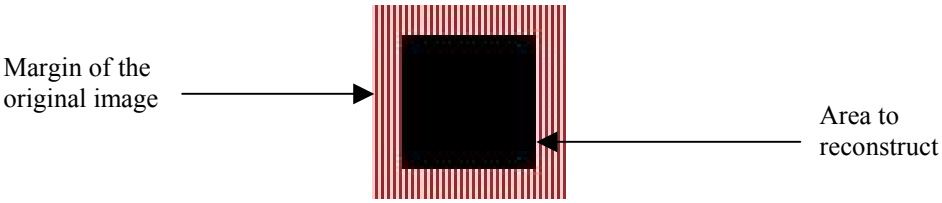


Figure 4.2.47 Intermediate step of the restoring method.

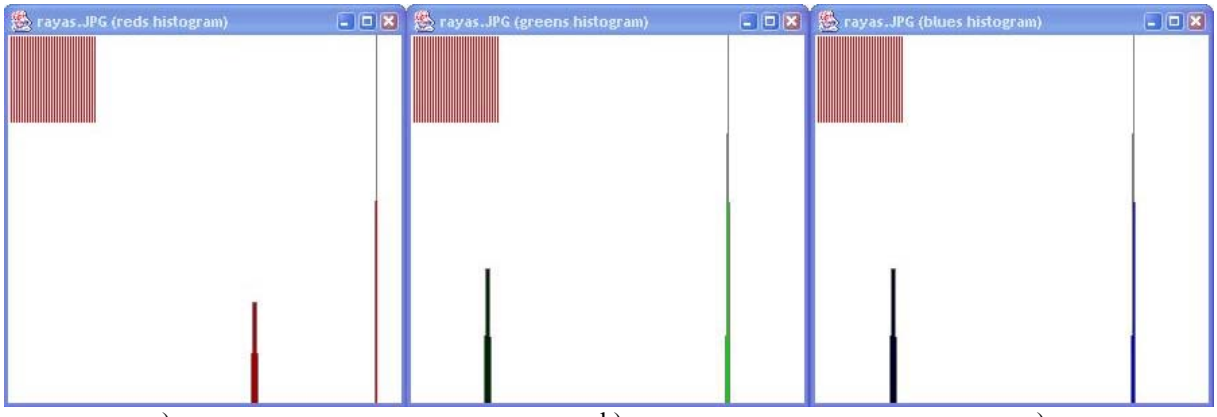


Figure 4.2.48 Histograms for the vertical sample. a)red histogram, b) green histogram and c) blue histogram.

After the restoring process, a perfectly reconstructed image is obtained, see figure 4.2.49.

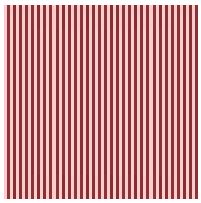
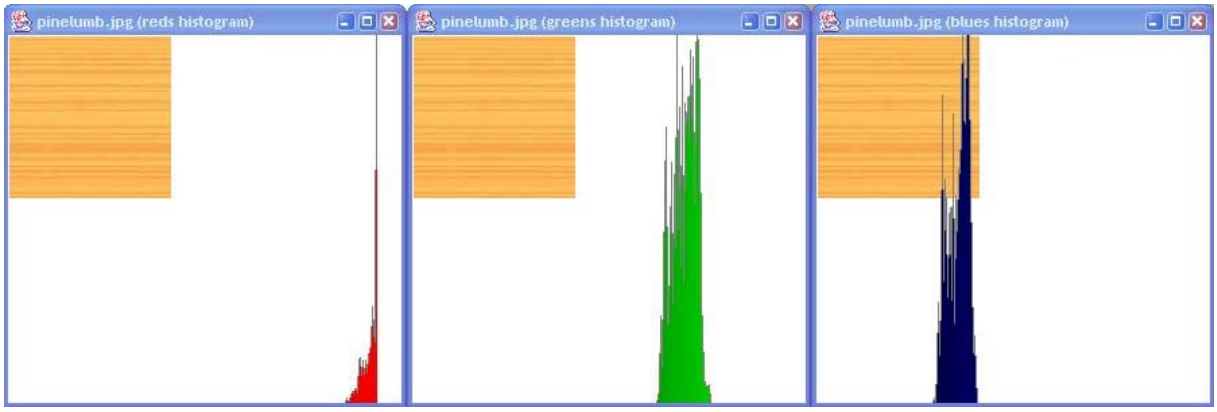


Figure 4.2.49 Restored vertical pattern with the second approach.

Once again, the original image is acquired. It can be said that regular and plain textures are perfectly reconstructed with both methods.

4.2.2.3.3 Third Category: Patterned Textures

The next type of texture is the wooden patterned image from the first experiment. The initial working area is 2 pixels high and 2 pixels wide. Histograms for the "pinelumb" sample are shown in figure 4.2.50.



a) b) c)

Figure 4.2.50 Histograms for the "pinelumb" sample. a)red channel, b) green channel and c) blue channel.

The analysis of the histogram is already described in the first experiment; just as a summary, the red values are on the bright side of the histogram, 255, which makes its distribution function narrow.

The green and blue values lie also in a small section of the histogram; that is why their distribution functions are also narrow. Next figure shows the result obtained in the second experiment, the image of the first experiment and the original sample.

a)
First experiment
(lower right sector
working area)

b)
Original sample

c)
Second experiment
(four sectors
working area)

Figure 4.2.51 Image results a) first approach, c) second approach and b) the original image.

To the left is the image from the previous experiment, whose co-occurrence matrices are calculated considering only the lower right sector of the observed pixel. In the centre appears the original image, to take it as reference. To the right side is the image reconstructed according to the second experiment, where the co-occurrence matrices are calculated with the whole four sections surrounding the reference pixel (upper right, upper left, lower right and lower left).

Both images are reconstructed with the same working area, 2x2 pixels. The results are obviously better in the second experiment, because the co-occurrence matrices contain more information than the ones in the first experiment.

The next comparison is between the two methods but this time with a working area of 3x3 pixels. The next figure shows the results.

a) First experiment (lower right sector working area)

b) Original sample

c) Second experiment (four sectors working area)

Figure 4.2.52 Comparison between the first and second approach with working area of 3x3 pixels, in the middle, the original image.

On the left side is shown the image obtained in the first experiment with working area of 3x3. On the right side is the image reconstructed in the second experiment, considering the whole four sectors surrounding the pivot pixel. In the middle lies the original texture.

In this example the first experiment shows better performance than the second because the amount of information is large enough to reconstruct it and display some patterns, while in the second experiment, the amount of information is so large, that is inexpedient and the image loses the slight pattern that is already achieved with the 2x2 working area. The next figure shows the result of the second experiment with even a wider working area, that is, 4x4 pixels.

a) First experiment (lower right sector working area)

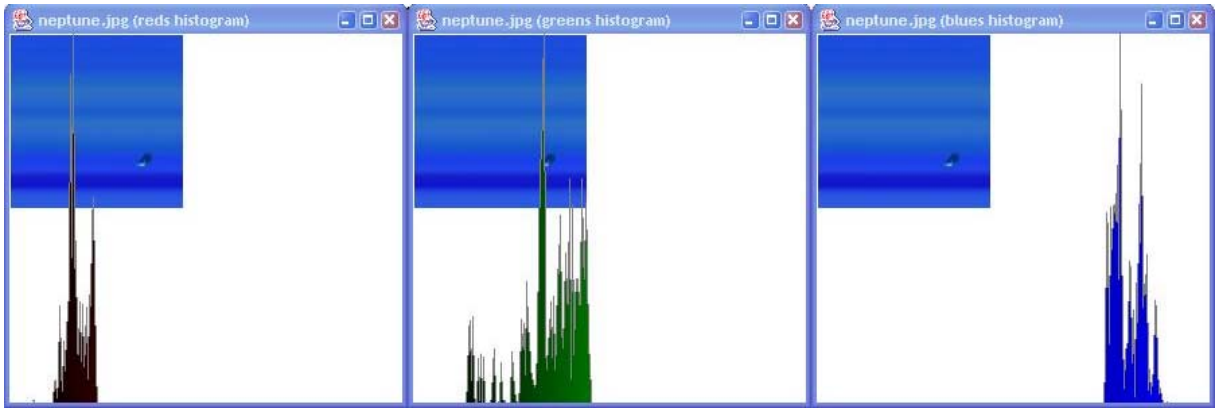
b) Original sample

c) Second experiment (four sectors working area)

Figure 4.2.53 Comparison between the first and second approach, in the middle, the original image.

To the left side is the image of the first experiment, in the middle the original and to the right is the image obtained in the second approach. Both results are worse than the previous examples and the main reason is the excess of information. That is why in both images there are almost no traces of the pattern that was established in the latter try with working area of 3x3 pixels.

The next image to analyse is the surface of Neptune, which is already calculated in the first experiment. Histograms show the colour distribution of the images, the next image shows the colour histograms for this texture.



a) b) c)

Figure 4.2.54 Histograms for Neptune sample. a)red histogram, b) green histogram and c) blue histogram.

Figure 4.2.55 shows the images obtained in both experiments with a 2x2 pixels area.

a)
First experiment
(lower right sector
working area)

b)
Original sample

c)
Second experiment
(four sectors
working area)

Figure 4.2.55 Comparison between the first and second approach, in the middle, the original image.

The left image is slightly better than the right one because its colour distribution is more similar to the original. In the right image is observed that the black spot was eradicated and filled with a solid texture, but the filling differs from the margin of the original picture. The next test runs with a working area of 3x3 pixels. Results are shown in figure 4.2.56a.

a)
First experiment
(lower right sector
working area)

b)
Original sample

c)
Second experiment
(four sectors
working area)

Figure 4.2.56 Results obtained in a) first approach, c) second approach and b) the original image.

The verdict between both images is clear, the first method reconstructs in a better way the image; the irregularity is covered and the pattern is very similar to the original, while the second experiment throws almost a copy of the previous result (figure 4.2.55c). The last try has a working area of 4x4 pixels. The result is shown in figure 4.2.57a.

a)
First experiment
(lower right sector
working area)

b)
Original sample

c)
Second experiment
(four sectors
working area)

Figure 4.2.57 Results obtained in a) first approach, c) second approach and b) the original image.

In this example is not easy to decide which method gives the best results, since the overflow of information produces almost homogeneous images. The main conclusion is that both methods, with a window of 4x4 pixels, do not give satisfactory results, at least with this texture.

The next image to test is the brick sample of figure 4.2.58. The complexity increases, comparing it to the Neptune sample. It has small irregularities along the image, and the joints are also a difficulty when restoring. Histograms in figure 4.2.59 show a better idea of this sample colour distribution.

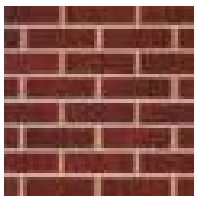


Figure 4.2.58 Brick sample.

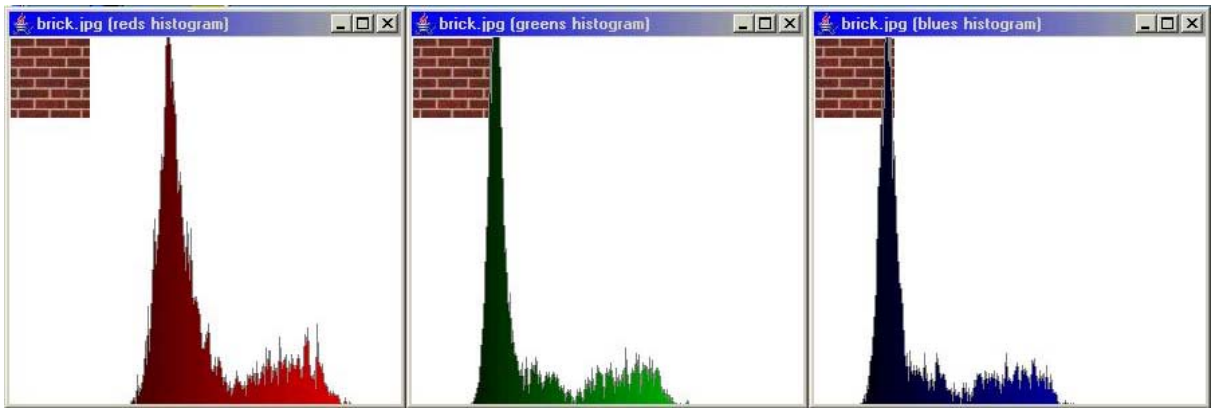
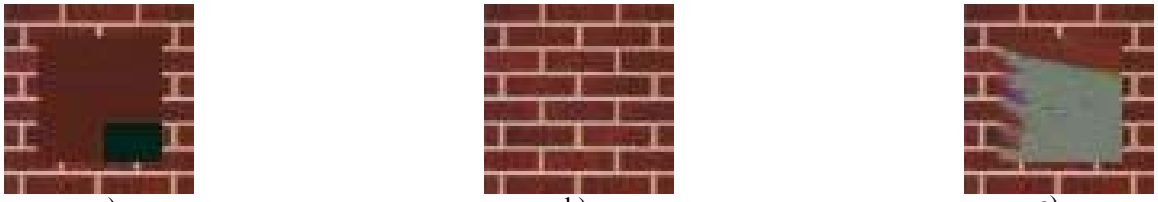


Figure 4.2.59 Histograms for brick sample. a)red histogram, b) green histogram and c) blue histogram.

Its histograms are very similar and cover almost the same range of values. Red channel is dominant and brighter than the other two. The working area for this experiment is 2x2 pixels wide. Figure 4.2.60c shows the reconstructed image.



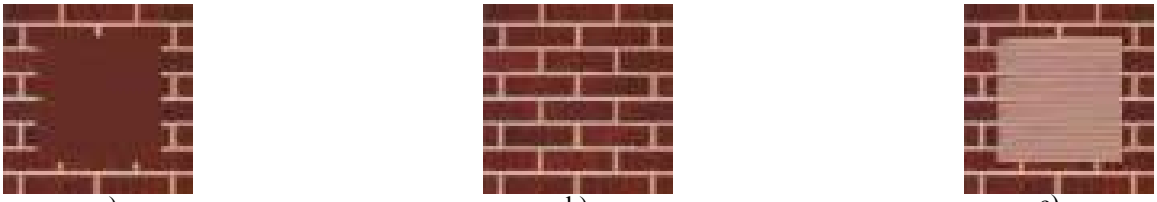
a) First experiment (lower right sector working area)

b) Original sample

c) Second experiment (four sectors working area)

Figure 4.2.60 Results obtained in a) first approach, c) second approach and b) the original image.

As observed, in the second experiment the texture starts to follow the original brick pattern, but diminishes and ends with a solid colour that differs from the dominant shade of the image. Next try has working area of 3x3 pixels. Figure 4.2.61c shows the results.



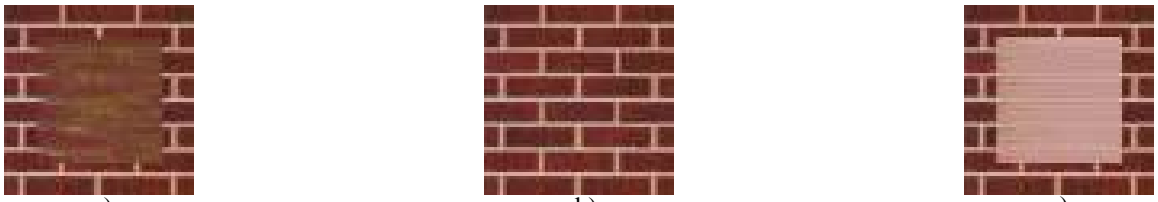
a) First experiment (lower right sector working area)

b) Original sample

c) Second experiment (four sectors working area)

Figure 4.2.61 Results obtained in a) first approach, c) second approach and b) the original image.

The second approach shows an excess of information, since the filling corresponds to the bricks joints. The result is not satisfactory enough. Next procedure runs with a working area of 4x4 pixels. Figure 4.2.62c shows the results.



a) First experiment (lower right sector working area)

b) Original sample

c) Second experiment (four sectors working area)

Figure 4.2.62 Comparison between the first and second approach, in the middle, the original image.

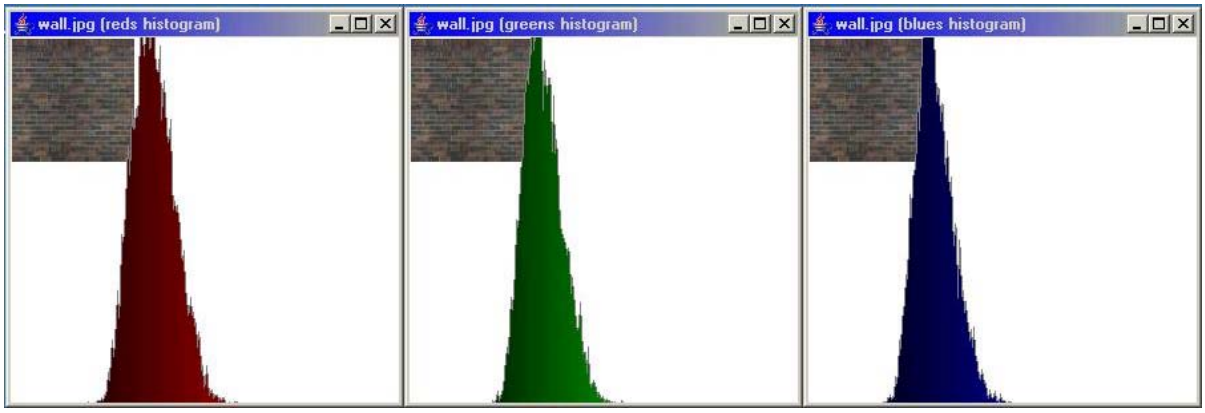
The first approach shows better reconstruction than the second experiment. However, the second has more information in co-occurrence matrices, it needs a filter that selects only the important data when reconstruct the image. Next chapter (4.2.3) handles this filter.

4.2.2.3.4 Fourth category: Irregular Patterned Textures

The next image to analyse is a wall sample. It is not regular anymore since there are strong jumps from dark shades to bright ones and *viceversa*. Figure 4.2.63 displays this sample.

Figure 4.2.63 Wall sample.

Its histograms are shown in next figure. They give a better idea of its colour distribution and also the difficulty when restoring the sample.



a) b) c)

Figure 4.2.64 Histograms for the wall sample. a)red histogram, b) green histogram and c) blue histogram.

Its colour distribution is similar in range and spectrum. The first test has a working area of 2x2 pixels and margin=25 pixels. The result is shown in figure 4.2.65c.

a)
First experiment
(lower right sector
working area)

b)
Original sample

c)
Second experiment
(four sectors
working area)

Figure 4.2.65 Comparison between the first and second approach, in the middle, the original image.

The second approach shows better results than the first one. It is one solid colour filling, but it nears to the dominant shade of the image. The next test is calculated with a 3x3 pixels working area. The margin of 25 pixels remains constant for this texture. Figure 4.2.66c shows the results.

a)
First experiment
(lower right sector
working area)

b)
Original sample

c)
Second experiment
(four sectors
working area)

Figure 4.2.66 Comparison between the first and second approach, in the middle, the original image.

The reconstruction of second approach starts to trace a pattern shape on the left side, but it fades away and the rest is filled with a solid colour. The last test for this sample runs with a working area of 4x4 pixels and the results are shown in figure 4.2.67c.

a)
First experiment
(lower right sector
working area)

b)
Original sample

c)
Second experiment
(four sectors
working area)

Figure 4.2.67 Results obtained in a) first approach, c) second approach and b) the original image.

The pattern is traced in both experiments, but the second reconstruction (figure 4.2.67c) shows them not only on the left side, but also in the upper section. However there is still not satisfactory. Further analysis is explained in section 4.2.3 of this chapter.

The last texture to test is a Jupiter sample, displayed in figure 4.2.68.

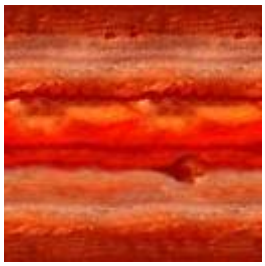
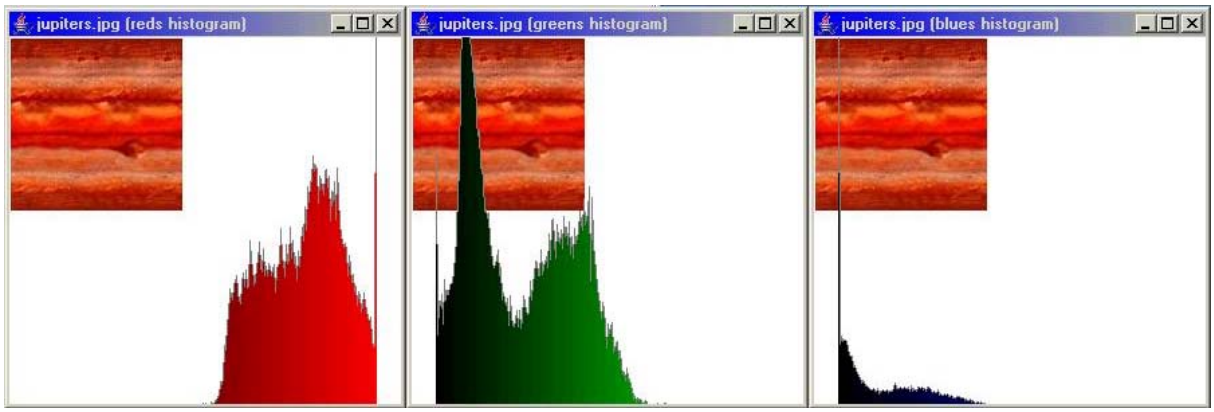


Figure 4.2.68 Jupiter sample.

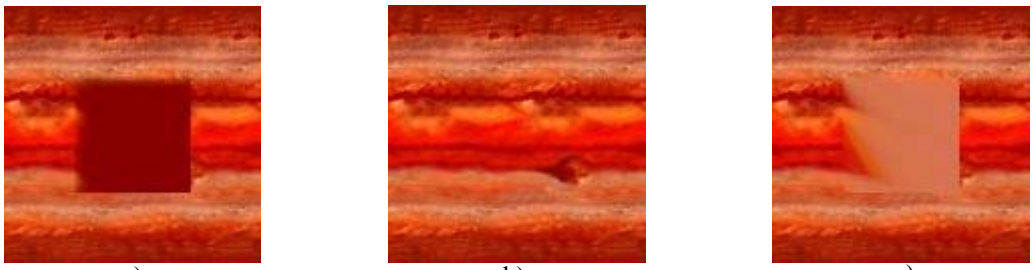
This sample is irregular, there are lots of shades and textures within the image, so its reconstruction can be more difficult to achieve. Its colour distribution is shown in next figure.



a) b) c)

Figure 4.2.69 Histograms for Jupiter sample. a)red histogram, b) green histogram and c) blue histogram.

As the histograms show, the colour distribution is very different from one channel to the others. Red is the dominant colour but also the more uniform; green has even a wider range and more peaks (maximums) and blue is barely present in this image. The first test is done with a working window of 2x2 pixels and a margin set to 35 pixels, which remains constant for this texture. Its results are shown in figure 4.2.70c.



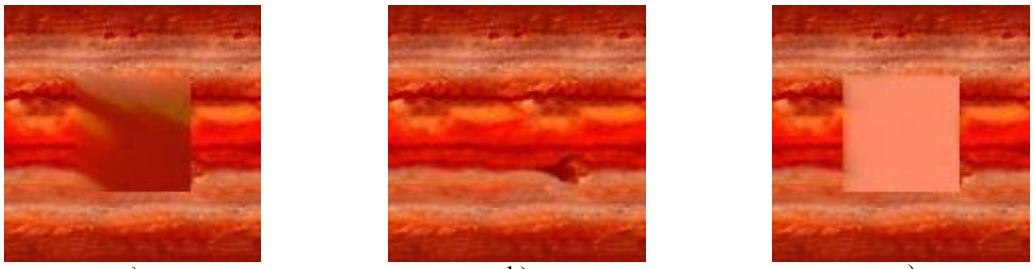
a) First experiment (lower right sector working area)

b) Original sample

c) Second experiment (four sectors working area)

Figure 4.2.70 Results obtained in a) first approach, c) second approach and b) the original image.

The second approach (figure 4.2.70c) shows better results. It starts to describe a pattern, but it fades away. This leads to the second try with a wider working area (3x3 pixels) to increase the quality of the reconstruction. Its results are shown in figure 4.2.71c.



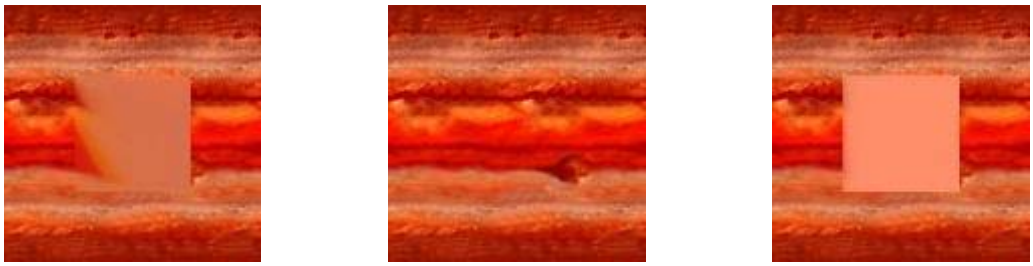
a) First experiment (lower right sector working area)

b) Original sample

c) Second experiment (four sectors working area)

Figure 4.2.71 Results obtained in a) first approach, c) second approach and b) the original image.

As observed, the 3x3 pixels working area causes a decreasing in the image quality. The pattern gets lost and the filling colour is too bright. In the first approach, this working area starts to trace some patterns. The last test for this sample is calculated with a 4x4 working area. The results are shown in figure 4.2.72c.



a) First experiment (lower right sector working area)

b) Original sample

c) Second experiment (four sectors working area)

Figure 4.2.73 Comparison between the first and second approach, in the middle, the original image.

The second approach has a clear overflow of data that causes the quality to diminish. This leads to the quest of a filter, able to select only the important information. This filter is further explained in next section (4.2.3).

4.2.3 Third approach: Wide Working Area and Entropy Filters

4.2.3.1 Acquiring Information

After evaluating the results of both methods, it is clear that the amount of information collected in the second method is larger than in the first one, but this is not improving the results, on the contrary, is decreasing the quality of the obtained image due to the overflow of information. Taking this into consideration, it should be possible to establish a filter that only allow the significant information to be considered during the learning and restoration processes.

It leads to the last co-occurrence matrices method analysed in this thesis. The working area remains constant as in the second experiment, the reason is that the significant pixels or colours are not only located in the lower right sector, but surrounding the pivot pixel. The filter that decides which information is necessary is the entropy.

This is an interesting feature that can be extracted from the co-occurrence matrices once they are normalised. In the area of computer graphics, the entropy is considered as a way to measure the chaos in the picture, *videlicet*, how unorganized is the image; this is not so distant from the original definition in thermodynamics: "a measure of the unavailable energy in a closed thermodynamic system that is also usually considered to be a measure of the system's disorder" (taken from Webster dictionaries).

The smaller the entropy, the more ordered is the image, hence the easier to process it and reconstruct it. For example, a solid colour image has entropy equal to zero, due the image is perfectly ordered.

Returning to the program, the matrices for this new experiment remain like the ones in the second experiment:

```
float smred [ver*2-1][hor*2-1][256][256];
float smgreen [ver*2-1][hor*2-1] [256][256];
float smblue [ver*2-1][hor*2-1] [256][256];
```

Where *ver*2-1* and *hor*2-1* are defined by the size of the working area and the next parameters contain the colour values, this is from 0 to 255.

The procedure to calculate the co-occurrence matrices is the same as in the second approach (see algorithm 4.2.5). The working area covers the eight different directions from the pivot pixel, but multiplied by the values "ver" and "hor", as shown in figure 4.2.38.

The method to calculate the matrices is exactly the same as in the previous experiments; the working area moves along the picture (limited by the specified margin) and takes the colour components to store them in the co-occurrence matrices. The code is the same as displayed in algorithm 4.2.5.

After the matrices are calculated and organized (see figure 4.2.39), they are normalised; each matrix is normalised separately applying the following equation.

$$NCM = \frac{CM}{\sum_{i=0}^{255}\sum_{j=0}^{255} v_{(i,j)}}$$

"NCM" represents the normalised co-occurrence matrix, "CM" is the original matrix and "v" represents each cell value of the matrix with indices "i" and "j".

After the matrices are normalised, the entropy is calculated; the next equation shows how to obtain the entropy from the co-occurrence matrices.

$$H(NCM) = -\frac{1}{256}\sum_{i=0}^{255}\sum_{j=0}^{255} v_{(i,j)} \ln v_{(i,j)}$$

Where "H(NCM)" represents the entropy of the normalised matrix, "v" represents each pixel value of the co-occurrence matrix, "i" and "j" are the indices.

When calculating the entropies, the procedure throws one entropy value from each co-occurrence matrix, which results in a matrix of entropies of (ver*2-1, hor*2-1) size. Figure 4.2.74 shows an example of an entropy matrix. Once the results are obtained, they are compared to each other; the minimum values are the most suitable to be chosen when reconstructing the image.

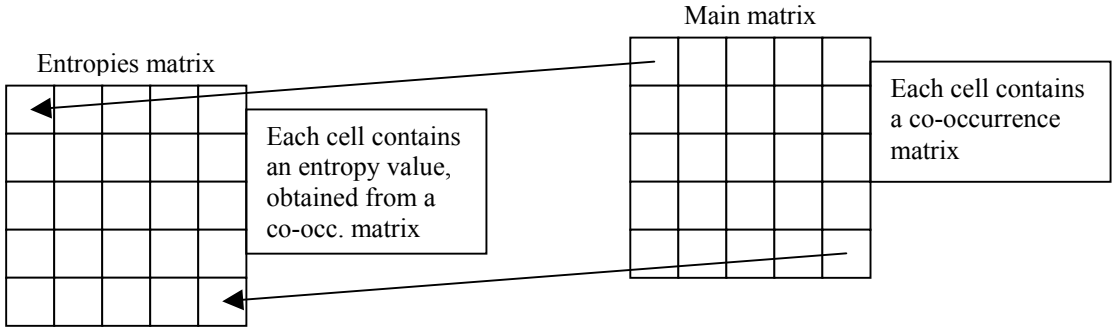


Figure 4.2.74 From the co-occurrence matrices stored in each cell of the main matrix, an entropy value is calculated and stored in the corresponding cell of the entropies matrix.

Once the entropies are stored in their respective matrix, it is necessary to make the co-occurrence matrices cumulative and normalise them again to obtain the distribution functions.

Until here, only the acquiring information process is described, which is exactly the same as in the second approach (see chap. 4.2.2.1). The second stage of the experiment is the reconstruction of the samples, but this time including the entropy filter.

4.2.3.2 Image Reconstruction

Like in the previous two experiments, the reconstruction does not start from a blank image, but from the margin of the original sample. It is necessary to take some information of the margin to compare it with the values stored in the co-occurrence matrices; this way it is possible to select the appropriate colours to fill the image in the most similar way to the original sample.

An introductory description of this process is shown in figure 4.2.40. In summary, the working area takes some pixels from the margin of the original sample, compares them with the distribution functions and chooses the most similar colour shade to generate the new pixel and fill the image.

The first part of the reconstruction is the creation of distribution functions, obtained from the co-occurrence matrices. This process is defined in algorithm 4.2.7, which is very similar to algorithm 4.2.6 but the difference is the implemented entropy filter.

Algorithm 4.2.7 Creation of distribution curves with entropy filters.

```
float Vred[] = new float[256];
float Vgreen[] = new float[256];
float Vblue[] = new float[256];
For k=0 to 255
{
  Vred[k]=1;Vgreen[k]=1;Vblue[k]=1;
}
```

```
For i=0 to ver
 For j=0 to hor
   disti=ver-1-i; distj=hor-1-j;
   For k=0 to 255
   {
/**************** first sector comparison**************/
    if Entred[i][j] < maxentred
     Vred[k]*=smred[i][j][red of newImage (p-disti, q-distj)][k];
    if Entgreen[i][j] < maxentgreen
     Vgreen[k]*=smgreen[i][j][green of newImage (p-disti, q-distj)][k];
    if Entblue[i][j] < maxentblue
     Vblue[k]*=smblue[i][j][blue of newImage (p-disti, q-distj)][k];
/**************** second sector comparison**************/
    if Entred[i][2*hor-j] < maxentred
     Vred[k]*=smred[i][ 2*hor-j][red of newImage (p-disti, q-distj)][k];
    if Entgreen[i][ 2*hor-j] < maxentgreen
     Vgreen[k]*=smgreen[i][ 2*hor-j][green of newImage (p-disti, q-distj)][k];
    if Entblue[i][ 2*hor-j] < maxentblue
     Vblue[k]*=smblue[i][ 2*hor-j][blue of newImage (p-disti, q-distj)][k];
/**************** third sector comparison**************/
    if Entred[2*ver-i][j] < maxentred
     Vred[k]*=smred[2*ver-i][j][red of newImage (p-disti, q-distj)][k];
    if Entgreen[2*ver-i][j] < maxentgreen
     Vgreen[k]*=smgreen[2*ver-i][j][green of newImage (p-disti, q-distj)][k];
    if Entblue[2*ver-i][j] < maxentblue
     Vblue[k]*=smblue[2*ver-i][j][blue of newImage (p-disti, q-distj)][k];
/**************** fourth sector comparison**************/
    if Entred[2*ver-i][2*hor-j] < maxentred
     Vred[k]*=smred[2*ver-i][ 2*hor-j][red of newImage (p-disti, q-distj)][k];
    if Entgreen[2*ver-i][ 2*hor-j] < maxentgreen
     Vgreen[k]*=smgreen[2*ver-i][ 2*hor-j][green of newImage (p-disti, q-distj)][k];
    if Entblue[2*ver-i][ 2*hor-j] < maxentblue
     Vblue[k]*=smblue[2*ver-i][2*hor- j][blue of newImage (p-disti, q-distj)][k];
  }
```

The difference between algorithms 4.2.6 and 4.2.7 is the condition "if Entropy value < maximum entropy value", because this is the filter that only accepts the co-occurrence values smaller than the entropy limit. Each colour is independent and usually, their entropy limits are different from each other. It is also very rare that two images have the same entropy filters.

As observed in figure 4.2.74, the matrix of entropies is a mirror of the matrix of co-occurrence matrices. From each co-occurrence matrix, an entropy value is obtained and placed in the same position in the main matrix. For example, if the red entropy value of the cell (2,2) is bigger than the limit "maxentred", the values of the red co-occurrence matrix located in the cell (2,2) (in the main matrix) are not considered when creating the distribution functions. This entropy filter blocks all the unnecessary information.

Once the entropy condition is fulfilled, the process continues exactly as in the previous two experiments. It takes the colour values of the pixel located in the position (p-disti, q-distj) and compares them to the ones stored in the first co-occurrence matrix, located in the position (i,j) of the main matrix. This goes on until it covers the 256 possible shades for each colour.

The distribution functions “Vred”, “Vgreen”, “Vblue” are obtained by multiplying the values of the co-occurrence matrices. The same procedure applies for the second co-occurrence matrix and it continues until it covers all the matrices. It is also necessary to make the arrays cumulative and normalise them. After finishing the distribution functions are obtained. Figure 4.2.41 shows an example of distribution curves.

After calculating the distribution functions, it follows the regeneration of the image. The procedures for the filling and reconstruction of the new image are already displayed in algorithms 4.2.3 and 4.2.4, since they are neither influenced by the change in the size of the working area nor affected by the entropy filter when obtaining the distribution arrays.

4.2.3.3 Results

Now it is time to compare the results obtained with this entropy filter to the ones acquired in the previous experiments.

4.2.3.3.1 First Category: Plain Textures

The first try is done with a monochromatic texture (created with Microsoft Paint 5.1 ™), see figure 4.2.75a. The working area for this procedure has height=2 pixels and width=2 pixels and the margin is set to 10 pixels wide that remains constant for regular textures. Working with plain textures is easy, since the entropy for these images is zero, which means that it is perfectly ordered. Entropy filters are not necessary for these textures. Since the value 192 is the only option for each colour channel, the image is easy to restore to its original state, so at the end of the reconstruction, an identical copy of the original is obtained, see 4.2.75b.

a) b)

Figure 4.2.75 Monochromatic sample. a)Original sample, b) restored image in third approach.

4.2.3.3.2 Second Category: Regular Textures

The next image to try in the experiment is a vertical pattern (created with Microsoft Paint 5.1 ™), see figure 4.2.76a. The parameters for this test are height=2 pixels and width=2 pixels.

a) b)

Figure 4.2.76 Vertical pattern. a) Original sample, b) reconstructed image with entropy filters.

As the histograms of the vertical pattern show in figure 4.2.77, the distribution functions are simple, so the reconstruction is easy to make. The entropy of this image is also small. After the restoring process a perfectly reconstructed image is acquired, see figure 4.2.76b.

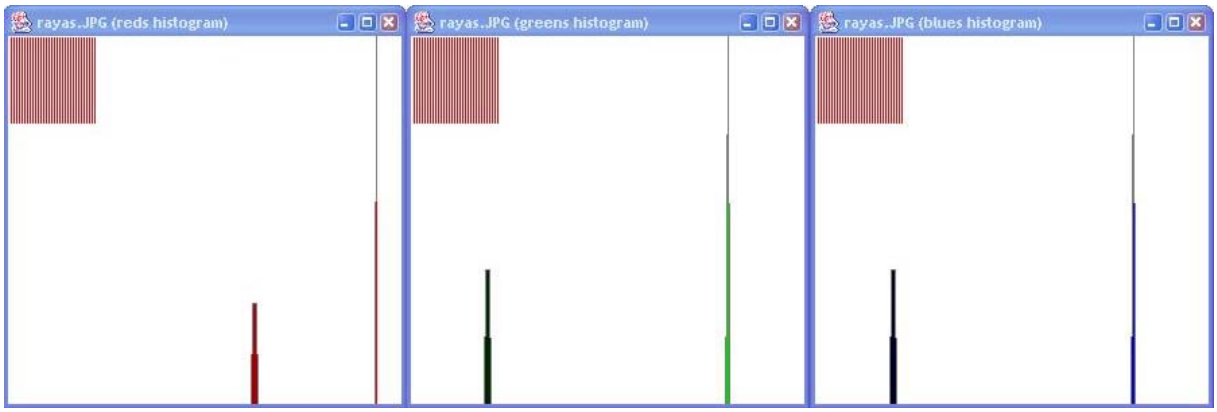


Figure 4.2.77 Histograms for the vertical sample. a)red histogram, b) green histogram and c) blue histogram.

The original image was perfectly reconstructed, and it still not necessary to establish an entropy filter due to the simplicity of the distribution functions. After this experiment a conclusion is reached. Plain textures and regular textures can be rebuilt with the help of co-occurrence matrices, with or without entropy filters.

4.2.3.3.3 Third Category: Patterned textures

The next texture to analyse is the "pinelumb" sample from the previous experiments, shown in figure 4.2.78. Its colour distribution is displayed with its histograms in figure 4.2.79.

Figure 4.2.78 "Pinelumb" sample.

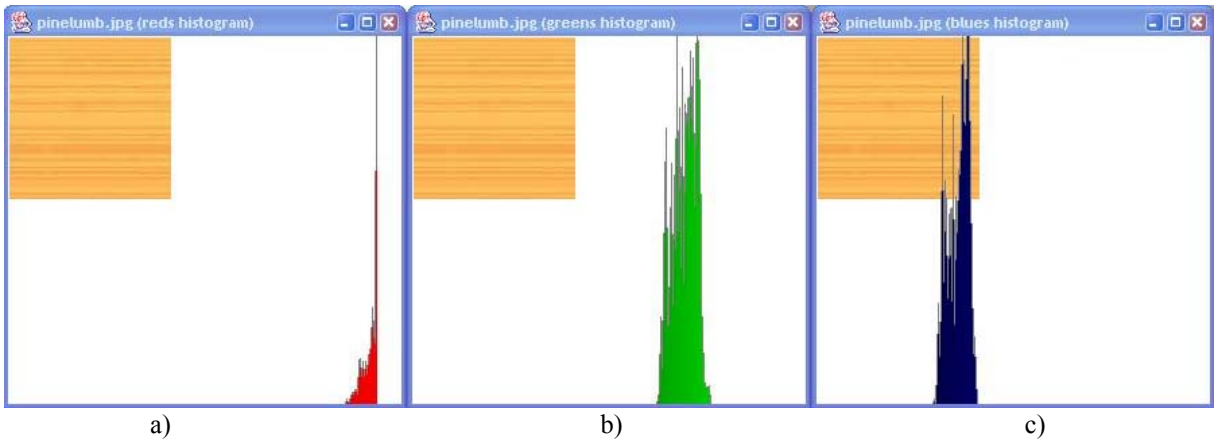


Figure 4.2.79 Histograms for "pinelumb" sample. a)red histogram, b) green histogram and c) blue histogram.

The following parameters are considered for its reconstruction: height=2 pixels and width=2 pixels. The entropy filters "maxent" are still not enabled because first it is necessary to obtain the entropy values for this image.

The generated image without the entropy filter is the one obtained in the second experiment, but the program also calculates the entropies for this image, which are displayed in the next table:

Table 4.2.1 Entropies for "pinelumb" texture. The values are rounded to 3 decimals.

	Lowest entropy	Highest entropy
Red channel	0.011	0.018
Green channel	0.015	0.023
Blue channel	0.013	0.022

Once the entropies are calculated a limit can be established, the variable "maxent". Since this limit only allows the lowest entropies to be taken for the distribution functions. For this sample it is set to 0.012 for the red channel, 0.016 for the green channel and 0.014 for the blue channel. The following pictures show the comparison of this new acquired image with the previous experiments results as well as with the original sample.

a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.80 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

The result obtained with the entropy filter is excellent, it is almost an exact copy of the original picture, there are only slight differences that at first view can not be noticed. Figure 4.2.81 shows the results with a working area of 3x3 pixels and the same entropy limits.

a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.81 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

With a working area of 3x3 pixels, the original image is perfectly regenerated; there are no noticeable differences between the new reconstructed texture and the original picture.

The last comparison is with a working area of 4x4 pixels, although the last test was already a success. Entropy limits remain the same as in the last test. Figure 4.2.82 displays the four images comparing all the experiments and the real texture.

a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.82 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

There are no differences between the 3x3 pixels working window and the 4x4 pixels window. Both of them collect all the necessary information in the co-occurrence matrices, (which with the 2x2 pixels area was still not enough). Finally, the entropy filter selects only the most important values to rebuild the image; this is the reason why the last two images in this experiment are so similar to each other and also to the original picture.

If the sample has a small irregularity, it is removed and covered with the surrounding pattern. Figure 4.2.83 shows the "pinelumb" texture with a small black hole, to test the accuracy of this method.

Figure 4.2.83 "Pinelumb" pattern with irregularity.

The initial working area for the reconstruction is 3x3 pixels wide and the entropy limits are the same as in the previous test. After running the program, the result in figure 4.2.84 is obtained.

Figure 4.2.84 "Pinelumb" pattern after running the entropy filters program. The small irregularity was replaced with the original pattern of the sample.

Before the reconstruction, the whole centre of the image limited by the margin is erased. When restoring, it compares only the values of the margin with the ones in the co-occurrence matrices, and creates a new image, so, the small irregularity disappears once the program finishes.

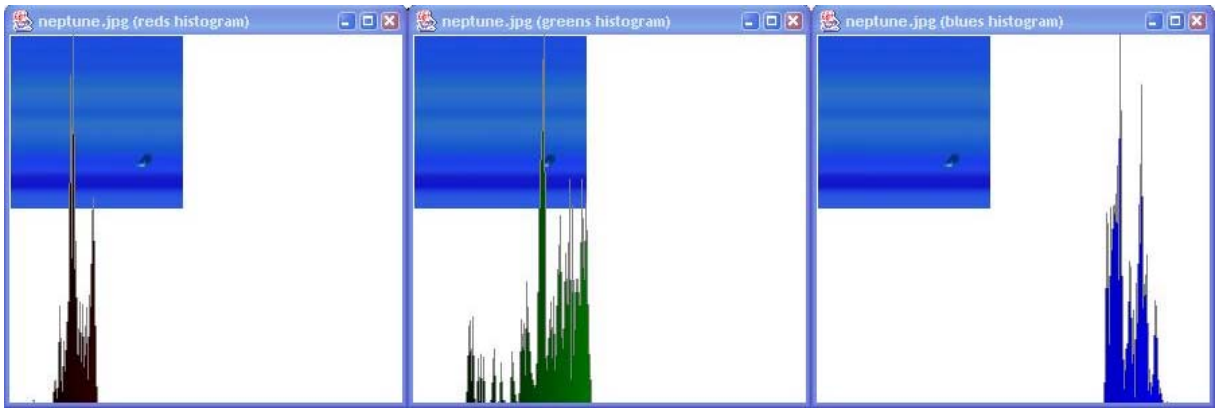
The next image to analyse is the Neptune sample, see figure 4.2.85, but first it is necessary to obtain its entropy values to be able to establish an appropriate filter. Table 4.2.2 shows the entropy values for the Neptune texture.

Figure 4.2.85 Neptune sample.

Table 4.2.2 Entropies for Neptune sample. The values are rounded to 3 decimals.

	Lowest entropy	Highest entropy
Red channel	0.014	0.019
Green channel	0.016	0.020
Blue channel	0.015	0.020

Figure 4.2.86 shows the Neptune histograms and the colour distributions.



a) b) c)

Figure 4.2.86 Histograms for Neptune sample. a)red histogram, b) green histogram and c) blue histogram.

Once the entropy limits are calculated, it is time to set the initial parameters for this sample. The working area is 2x2 pixels wide, and the entropy filters are, for the red channel 0.015, the green channel 0.018 and the blue channel 0.017. The results are displayed in figure 4.2.87.

a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.87 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

It is clear that the experiment with the entropy filter is far better than the previous experiments. The colour shades and patterns are almost the same as in the original picture, plus, the small irregularity on the lower right section of the image was eradicated and reconstructed with the appropriate pattern.

The next test has working area of 3x3 pixels and the entropy filters remain the same as in the previous test. These results and also its comparison with the previous two experiments are displayed in figure 4.2.88.

a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.88 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

There is almost no noticeable difference between the 2x2 window test and the 3x3 window obtained image, but it is slight better the result acquired in the 3x3 window. The irregularity is also corrected and the colour shades are even closer to the ones of the original sample. It is also obvious that the entropy filter method is far better than the previous. Last comparison has working area of 4x4 pixels. Entropy filters remain the same as in the last test. Figure 4.2.89 shows the results and their comparison.

a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.89 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

It is clear that the entropy filter is the best method to reconstruct images when using co-occurrence matrices. The main question is, which working area is the most suitable to rebuild the image. The 3x3 window as well as the 4x4 area are better options than the 2x2 working area. In this specific sample, the 3x3 window throws the most similar image to the original sample, although the differences are almost impossible to notice.

The next texture to analyse is a brick sample, see figure 4.2.90. It has more colour shades and the contrast between bricks and joints is very noticeable.

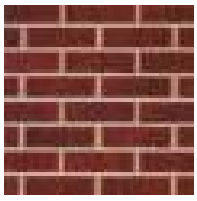
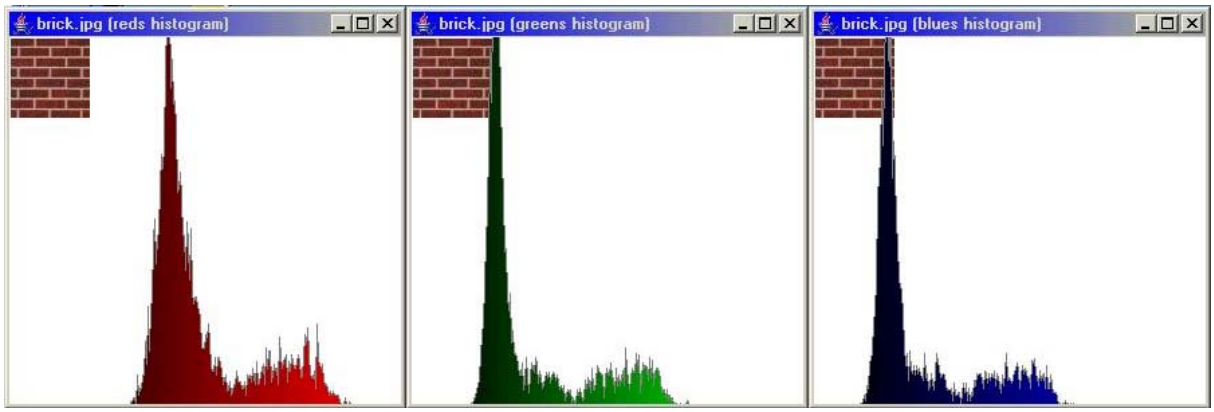


Figure 4.2.90 Bricks sample.

The colour distribution for this sample is shown in figure 4.2.91 with its histograms.



a) b) c)

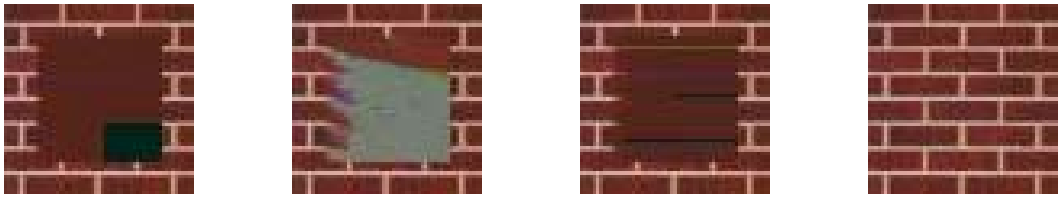
Figure 4.2.91 Histograms for the bricks sample. a)red histogram, b) green histogram and c) blue histogram.

After the program runs, its entropies are obtained. They are shown in table 4.2.3.

Table 4.2.3 Entropies for the bricks sample. The values are rounded to 4 decimals.

	Lowest entropy	Highest entropy
Red channel	0.0284	0.3020
Green channel	0.0273	0.0294
Blue channel	0.0275	0.0295

The initial parameters for the first test are height=2 pixels and width=2 pixels. The entropy limits for this sample are 0.0300 for the red channel, 0.0290 for the green channel and also for the blue channel. Figure 4.2.92c shows the results with entropy filter.



a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

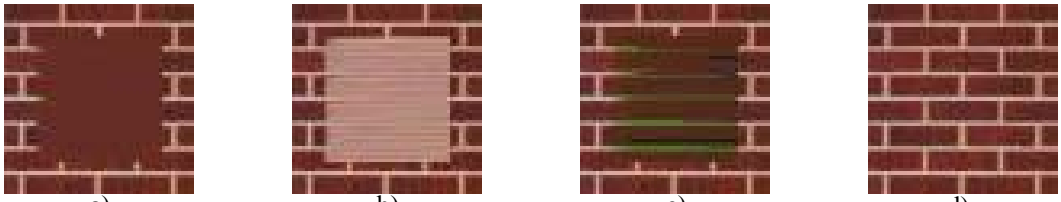
c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.92 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

As observed in previous figure, the result obtained with entropy filter is better than the other two approaches. It traces the joint and brick pattern in the upper part of the image.

The next analysis runs with a working area of 3x3 pixels. Its entropy limits remain the same as in the previous test. This approach results are shown in figure 4.2.93c.



a) First experiment (lower right sector working area)

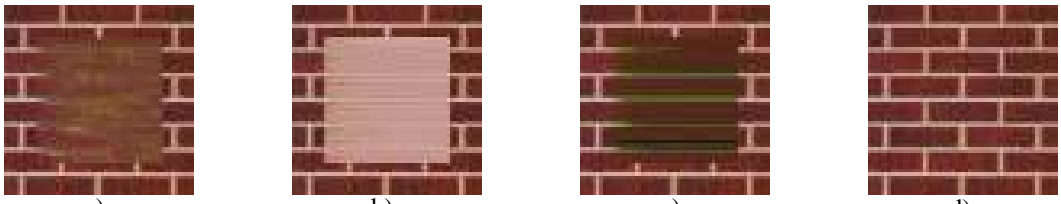
b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.93 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

The result obtained with the entropy filter increases in quality. The pattern is almost restored to its totality and the shades are also similar to the original sample. For the next test, a working area of 4x4 pixels is selected. The entropy limits remain intact. Its results are shown in figure 4.2.94c.



a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.94 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

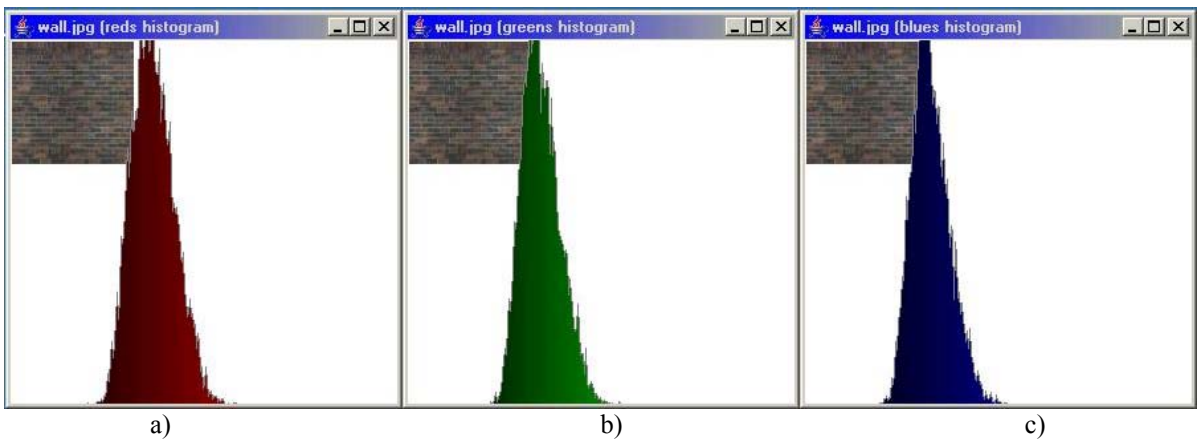
It is clear that the entropy filter is the best method for reconstruction, at least for this type of textures. The result obtained in the last test is even better than the previous with 3x3 pixels working window. Although is not a perfect reconstructed copy, the image is similar to the original. The reconstruction is not as good as the Neptune sample or the "pinelumb" texture because of the joints between the bricks.

4.2.3.3.4 Fourth Category: Irregular Patterned Textures

The next images to analyse are irregular and more complex. The first texture to analyse is a wall sample. It is an irregular image because of the changes between dark and bright areas. Figure 4.2.95 displays this sample.

Figure 4.2.95 Irregular wall sample.

The image histograms are displayed in figure 4.2.96. Its spectra and range of colour are very similar, but histograms are not a comparative measure to determine the complexity of the image, they only show its linear colour distribution.



a) b) c)

Figure 4.2.96 Histograms for the wall sample. a)red histogram, b) green histogram and c) blue histogram.

The entropy limits for this sample are obtained after the program runs. They are displayed in table 4.2.4.

Table 4.2.4 Entropies for the wall sample. The values are rounded to 4 decimals.

	Lowest entropy	Highest entropy
Red channel	0.0286	0.0310
Green channel	0.0281	0.0305
Blue channel	0.0280	0.0303

The working window for this experiment is 2x2 pixels wide and the margin is 25 pixels wide, which remains constant for this texture. As observed in table 4.2.4, the range of entropy values is very similar for the three colours, so, its entropy limits are the same, this is a value of 0.0300. This approach results as well as the reconstructed images of the previous experiments are displayed in figure 4.2.97.

a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.97 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

As observed, the best result is not easy to determine, even with the entropy filter, the image is filled with a solid colour. Similar to the second experiment, it begins to trace the bricks pattern, but it fades away. The third experiment colour nears to the dominant shade of the original sample. A wider working area is needed to reconstruct the wall pattern, so, the second test is calculated with working area of 3x3 pixels and entropy filters of 0.0300 for the each channel, like in the previous test. Its results are shown in figure 4.2.98c.

a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.98 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

The image obtained with the entropy filter traces longer image patterns than in the previous results. But it still not covers the entire image and fills it with a homogeneous colour.

The last test for this sample is calculated with even a wider working window of 4x4 pixels and the same entropy filters as the last test. Figure 4.2.99c displays the results obtained with this working area.

a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.99 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

As expected, the augment of the working area increases the quality of the reconstructed image. The pattern is traced almost to the middle of the image and then fades away. This is the best result obtained from this irregular sample, but the irregularities are very difficult to eliminate or reconstruct properly.

The following image is a Jupiter sample. It is also an irregular sample, and even more complex than the previous wall image. Figure 4.2.100 displays this image.

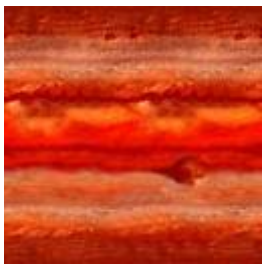
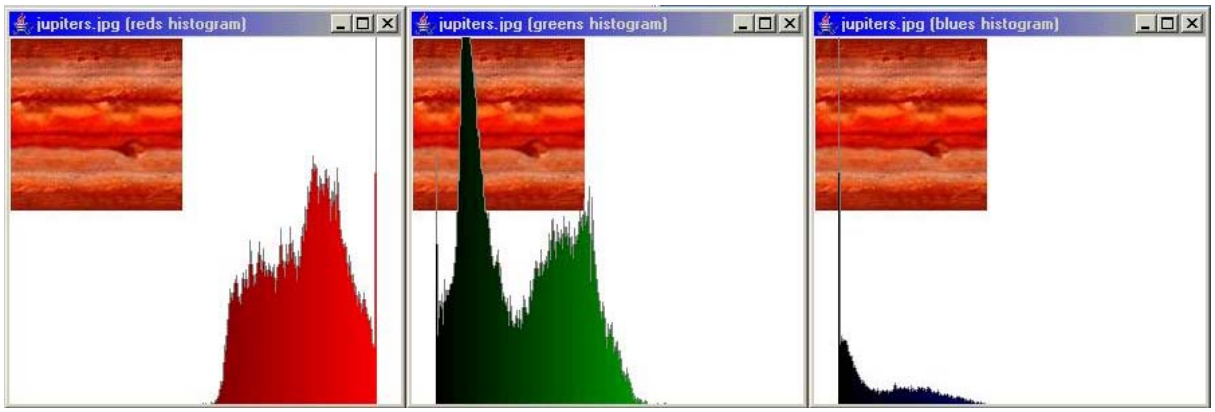


Figure 4.2.100 Sample of Jupiter surface.

As observed, there are some slight traces of horizontal patterns but they are not regular like the Neptune sample or the “pinelumb” sample. The colour and shape variations are even more drastically as in the wall sample.

As shown in the histograms, see figure 4.2.101, the colour distribution is completely different for each channel, and the spectra are not even similar, neither in shape nor in range.



a) b) c)

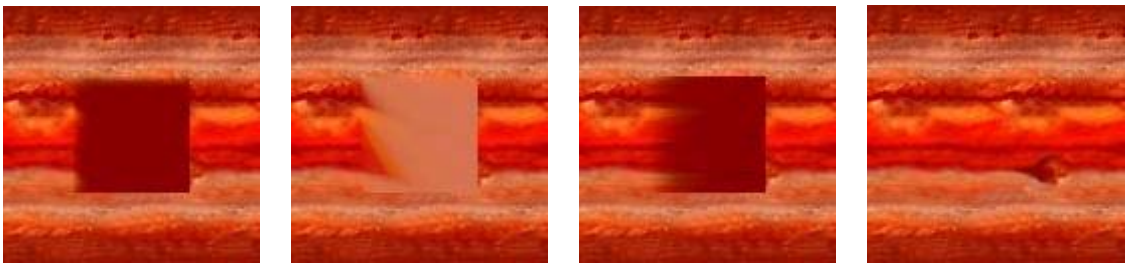
Figure 4.2.101 Histograms for the Jupiter sample. a)red histogram, b) green histogram and c) blue histogram.

The entropy values are obtained after the programs runs for the first time. The next table shows the entropy limits for this sample.

Table 4.2.5 Entropies for Jupiter sample. The values are rounded to 3 decimals.

	Lowest entropy	Highest entropy
Red channel	0.0290	0.0329
Green channel	0.0295	0.0339
Blue channel	0.0250	0.0285

The parameters to calculate this image are 2x2 pixels working area, margin of 35 pixels and entropy limits of 0.0310 for the red channel, 0.0320 for the green channel and 0.0270 for the blue channel. Its results are displayed in figure 4.2.102c.



a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

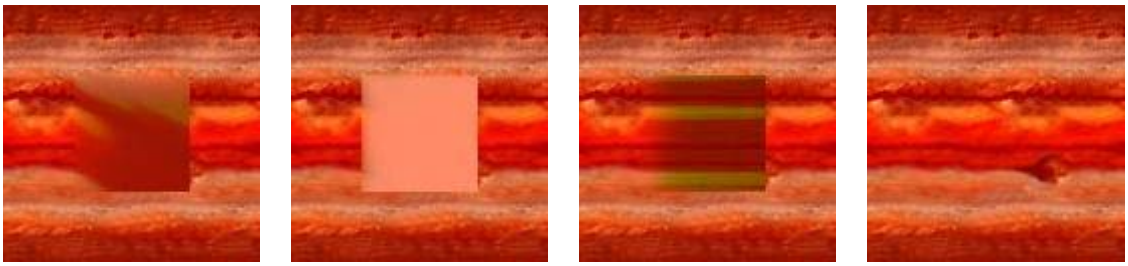
c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.102 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

As observed, the entropy filter experiment calculates a slight better reconstruction than the previous two experiments. There are some traces of the original pattern on the left side of the image, but these fade away. However, this experiment delivers an image that is not monochromatic anymore, and the colour shades are similar to the dominant colour of Jupiter.

For the next test, the working window is increased to 3x3 pixels. Entropy limits and margin are also the same as in previous test. Figure 4.2.103c shows the results obtained with these new parameters.



a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

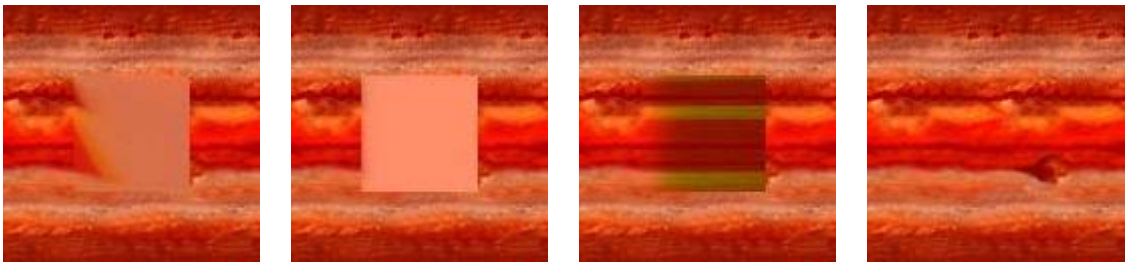
c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.103 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

The entropy filter delivers a patterned sample of Jupiter, it is not a solid colour anymore, but is still not the same as the original sample. There are too many irregularities and the procedure can not reconstruct them. It reconstructs a pattern based on the acquired co-occurrence matrices.

The last test for this sample runs with a wider working area of 4x4 pixels and the same entropy limits used in both previous tests. Figure 4.2.104c shows the results obtained with the entropy filter method.



a) First experiment (lower right sector working area)

b) Second experiment (four sectors working area)

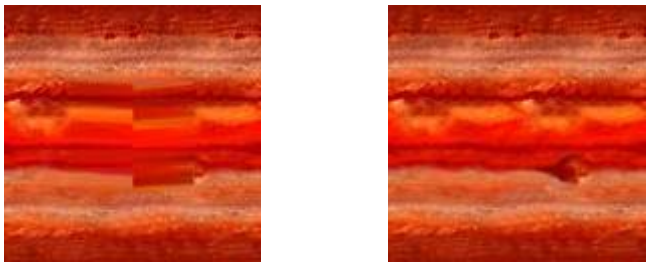
c) Third experiment (four sectors working area and entropy filter)

d) Original sample

Figure 4.2.104 a)Result of the first approach, b)result of the second experiment, c)result of the third approach and d)original sample.

The results obtained with the 4x4 pixels working area and entropy limits are very similar to the 3x3 pixels working window. The experiment traces also a regular pattern. Although this is not a copy of the original image, this image is better than the obtained in previous two experiments. The texture definition is not present, but the locations for the colour shades are similar to the original sample.

One last calculation is done with a wider working area of 5x5 pixels. Margin and entropy filters remain constant. Figure 4.2.105a shows this obtained result.



a)
Third experiment (four sectors working area and entropy filter)

b)
Original sample

Figure 4.2.105 a)result of the third approach and b)original sample.

This last result is by far, the best reconstruction obtained. It fills the erased area with regular pattern, but the colour shades are very similar to the Jupiter original sample.

After the comparison between the different approaches, a conclusion can be established. The use of co-occurrence matrices with the help of appropriate entropy filters is a suitable option to reconstruct regular patterned images and eliminate small irregularities from the original pictures.

For irregular patterns, this method is also appropriate, but only for the restoration of narrow areas (see fig.4.3.105a) since the method traces regular pattern to fill the desired area. The next chapter (4.3) reconstructs images with the help of vector quantization. This approach can be an alternative solution to the reconstruction of irregular pattern textures with co-occurrence matrices.

4.3 Learning Based on Vector Quantization

Vector quantization is a lossy data compression technique used in many applications such as image and voice compression and pattern recognition. A short and clear introduction to Vector Quantization applied to image compression is displayed below.

In Scalar Quantization the values are represented by a fixed subset of representative values. For example, having 16 bit values and sending only 8 most significant bits, an approximation of the original data is obtained at the expense of precision. In this case the fixed subset is all the 16-bit numbers divisible by 256, for example, 0, 256, 512,...

In Vector Quantization, values are not represented individually but (usually small) as arrays. A typical example is a colour map: a colour picture can be represented by a 2D array of triplets (RGB values). In most pictures those triplets do not cover the whole RGB space but tend to concentrate in certain areas. For example, the picture of a forest has, typically, a lot of green. A relatively small subset (typically 256 elements) of representative colors is selected, for example, RGB triplets. Next step approximates each triplet to the representative of that small set. In case of 256 1 byte can be used instead of 3 for each pixel [45].

Applied to texture reconstruction, the basis of this method is the obtaining of image information by initial vectors, which are trained later. Once the training is finished, the most suitable vectors (winners) are considered to restore the desired texture. More information about vector quantization can be obtained in [46, 47, 48, 49].

4.3.1 First approach: Vector Learning with Euclidean Distance

4.3.1.1 Obtaining Information

A different approach from the co-occurrence matrix is the use of vector quantization, which is explained in this part of the thesis. As observed in the previous method, (see chapter 4.2) regular textures can be learned and then reconstructed with the help of co-occurrence matrix approach (with satisfactory results). This current method goal is to obtain the same or better results than with the co-occurrence matrices.

4.3.1.1.1 Defining the Initial Vectors

The first step of this procedure defines the initial vectors, which are trained to learn the image most important features for a later reconstruction of the texture. The array where the vectors are stored is defined as follows.

```
float supvector[][][][]=new float [colour][size][dim][dim];
```

The first index corresponds to the colour, due to the fact that they are independently calculated; the number of vectors is given by the constant “size”. Their dimension is defined by the constant “dim” and is quadratic.

Vectors are chosen randomly, a sample of dimension “dim” is taken from the image, where its red, green and blue components are extracted and stored in the correspondent vectors (see algorithm 4.3.1).

Algorithm 4.3.1 Obtaining of initial vectors.

```
For z=0 to size{
 randver=(int)(Math.random() * (ih-dim));
 randhor=(int)(Math.random() * (iw-dim));
 For x=0 to dim
  For y=0 to dim
   morig2d[y][x]=pixels2d[y+randver][x+randhor];
 For x=0 to dim
  For y=0 to dim{
   supvector[red][z][y][x]=red of morig2d[y][x];
   supvector[green][z][y][x]=green of morig2d[y][x];
   supvector[blue][z][y][x]=blue of morig2d[y][x];
  }
 Vecdist[red][z][(int)(supvector[red][z][middle][middle])]+=1;
 Vecdist[green][z][(int)(supvector[green][z][middle][middle])]+=1;
 Vecdist[blue][z][(int)(supvector[blue][z][middle][middle])]+=1;
}
```

The main loop controls the number of learning vectors. The coordinates (“randhor”, “randver”) give the position of the initial pixel of the sample, then a square is traced from that position to the coordinates (“randhor+dim-1”, “randver+dim-1”) in the second loop. An example of this procedure is shown in figure 4.3.1.

The sample is temporary stored in the array “morig2d”. The following loop takes the colour components from that sample and stores them in the vector array “supvector”.

A distribution function "Vecdist" is needed to keep the colour shades statistics of the initial vectors. The first index represents the colour channels (red, green, blue); "size" corresponds to the number of initial vectors and "255" is the range of shades for each colour.

Distribution function follows the same principle of the histogram function; it contains the number of times each colour shade occurs when establishing the vectors. It takes the middle value of each initial vector, extracts the colour component and stores it to its correspondent allocation. After storing the information on the learning vectors, the second stage begins, this is, the learning process (vector training).

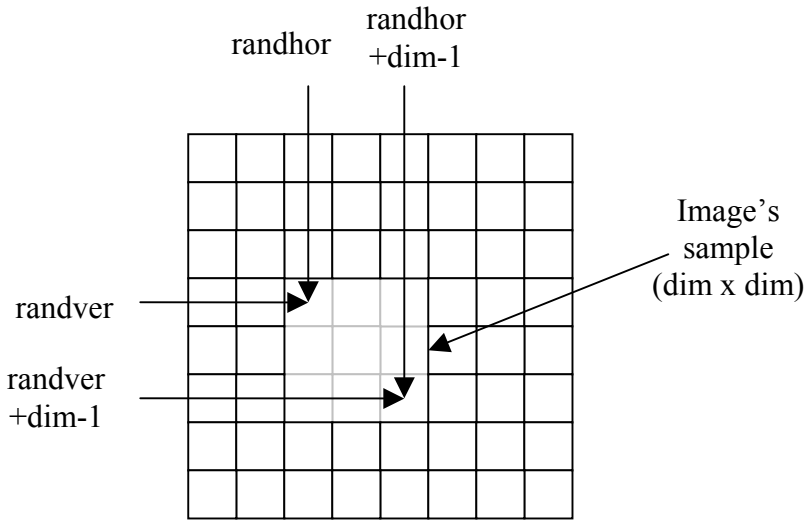


Figure 4.3.1 Random sample to store as initial vector.

4.3.1.1.2 Vector training

One random sample (comparative vector) of the image is taken; its colour components are extracted and are compared with each learning vector. The vector with which the comparative vector has the smaller distance is called the winner vector. It follows the principle of "winner takes all" since only the winner vector is trained in each epoch. The process runs until it reaches the specified amount of epochs. For more information about the learning process of vector quantization see [50].

First, it is necessary to calculate the distance between the single pixels of the vector area; the sum of them gives the total distance between the comparative vector and the learning vector. The distance between vectors is the Euclidean distance, which is the straight-line distance between two points, $p_1\ (x_1, y_1)$ and $p_2\ (x_2, y_2)$; its equation is the following:

$$Euclidean\ \ dist. = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Algorithm 4.3.2 shows the distance calculation between vectors.

Algorithm 4.3.2 Euclidean distance calculation.

```
For w=0 to epochs {
  takesample();
  For c=0 to size
   For i=0 to dim
    For j=0 to dim
```

```
{
    distauxr+=Square root of ((supvector[red][c][i][j]-veccompar[red][i][j])*
    (supvector[red][c][i][j]-veccompar[red][i][j]));
    distauxg+= Square root of ((supvector[green][c][i][j]-veccompar[green][i][j])*
    (supvector[green][c][i][j]-veccompar[green][i][j]));
    distauxb+= Square root of ((supvector[blue][c][i][j]-veccompar[blue][i][j])*
    (supvector[blue][c][i][j]-veccompar[blue][i][j]));
  }
}
```

As the code shows, the distance is calculated following the Euclidean distance equation, where the values are stored in the independent variables, "distauxr" for the reds, "distauxg" for the greens and "distauxb" for the blues.

The sample is compared to every learning vector, which is defined in the "size" loop. After finishing all the comparisons, the nearest vector to the sample is elected as the winner vector. The next step moves the winner vector in direction of the sample, which is given by the function *winner vector = alpha * compare vector + (1-alpha) * winner vector*, where alpha is defined as 1/epochs.

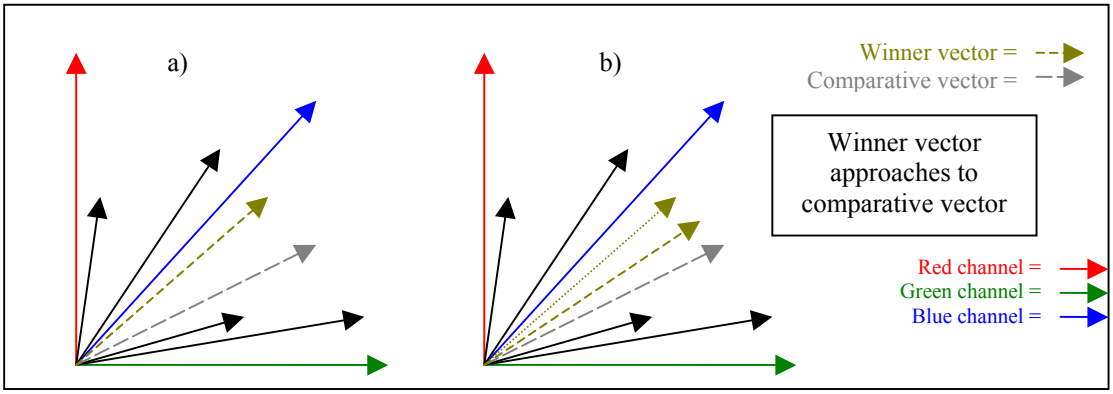


Figure 4.3.2 Vector training. a)Initial position, b)winner vector approaches to comparative vector.

The following step stores the information acquired with the initial vectors; they take the middle value of the comparative vector and store it to its correspondent allocation. Each colour channel is treated independently, as shown in following code.

Algorithm 4.3.3 Extraction of information from initial vectors.

```
Vecdist[red][winred][veccompar][red][middle][middle]]+=1;
Vecdist[green][wingreen][veccompar][green][middle][middle]]+=1;
Vecdist[blue][winblue][veccompar][blue][middle][middle]]+=1;
```

Algorithm 4.3.3 represents only the first epoch. The next epoch takes another random sample, compares it with all vectors, chooses the winner vector, moves it in direction of the recent sample and takes the colour values of the sample to store them in the distribution functions, an example of the vector training is displayed in figure 4.3.2.

The whole training is defined by the "epochs" loop. So, at the end, there is an "epochs-number" of winner vectors. Some of them are winners more than once and some other maybe not even once; this is randomly determined. Next step normalises the distribution functions.

The distribution functions have for each vector a histogram-like statistic. For each vector position index, a sum of its values is made and then every value is divided by this sum, this process is the normalisation of the functions.

4.3.1.2 Image Reconstruction

Until this part of the experiment, the processes acquire the statistics of the image; the next stage handles the construction of a new image based on the obtained information stored in the learning vectors. This process reconstructs the image by taking the information of the nearest pixels and comparing its values with the vector values. The procedure does not start from an empty image. It takes reference from the margin of the original picture and builds the new image according to that information.

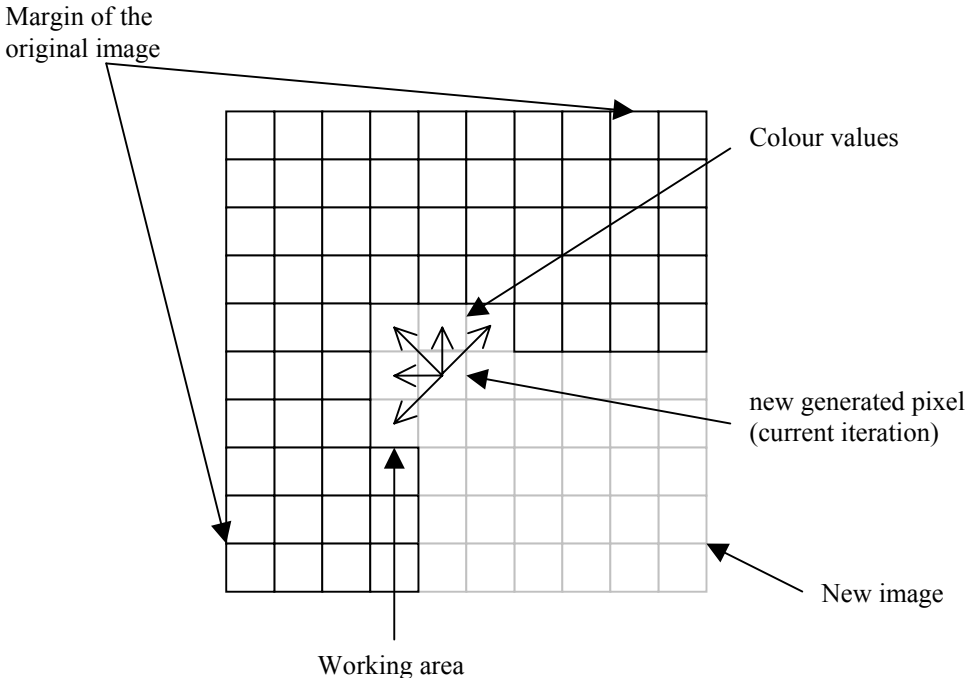


Figure 4.3.3 Reconstruction procedure.

Figure 4.3.3 displays the *modus operandi* of the reconstruction. The reconstruction order is vertical and the working area is determined by the constant "dist" as algorithm 4.3.5 shows; in figure 4.3.3, "dist" has the value of one. Each new generated pixel gathers information from its neighbours, and compares the acquired values to the ones stored in the learning vectors. The process is repeated until it reaches the last image pixel.

The first step creates three arrays, one for each colour, which contain the compressed information of the textures already acquired from the learning vectors. These arrays are 256 elements long and represent the possible values that each colour channel can choose to rebuild the image. Algorithm 4.3.4 displays this procedure.

Algorithm 4.3.4 Selection of winner vectors with Euclidean distance.

```
float Vred[] = new float[256];
float Vgreen[] = new float[256];
float Vblue[] = new float[256];
For c=0 to size
 For i=0 to dim
```

```
For j=0 to dim  {
  di=p-dist+i;dj=q-dist+j;
  if (di < p || ((di == p) && (dj < q)) || ((di > p) && (dj < p))
||di > imagewidth-margen-1 || dj > imageheight-margen-1)     {
    distauxr+= Square root of ((supvector[red][c][i][j] - red of newimg2d[di][dj])*
    (supvector[red][c][i][j] - red of newimg2d[di][dj]));
    distauxg+= Square root of ((supvector[green][c][i][j] -green of newimg2d[di][dj])*
    (supvector[green][c][i][j] - green of newimg2d [di][dj]));
    distauxb+= Square root of ((supvector[blue][c][i][j] - blue of newimg2d [di][dj])*
    (supvector[blue][c][i][j] - blue of newimg2d [di][dj])); } }
```

The main loop corresponds to the constant "size", this is, the quantity of learning vectors. The inner loop ("dim"x"dim") defines the working area with help of the "if" condition, which restrains the pixels taken to compare as shown in figure 5.3.3. Variables "p" and "q" represent the actual position of the pixel in the new image. "Distauxr", "distauxg" and "distauxb" determine the distance between the vectors and the working area. The winner vector is the one with the smallest distance towards the working area.

The functionality of this procedure is the same as in the learning stage; the difference is that the learning vectors "supvector[colour][c][i][j]" are not compared to a random sample, but to the working area, which is represented by "newimg2d[di][dj]". The process aims to determine a winner vector, from which the distribution function is taken to reconstruct the pixel at the current position as it shows algorithm 4.3.5.

Algorithm 4.3.5 Creation of distribution functions.

```
For i=0 to 255
  Vred[i]=Vecdist[red][winred][i];
For i=0 to 255
  Vgreen[i]=Vecdist[green][wingreen][i];
For i=0 to 255
  Vblue[i]=Vecdist[blue][winblue][i];
```

Since the three channels are independent, they are chosen according to the winner vectors of each colour. After calculating the distribution functions, it is necessary to make them cumulative and then normalise them. A loop from i=1 to 255 adds the values of the vector at the actual position (i) with the prior position (i-1). The process divides the whole vectors by the value in the last vector position (255) and finishes the normalisation. Figure 4.3.4 shows an example of distribution functions.

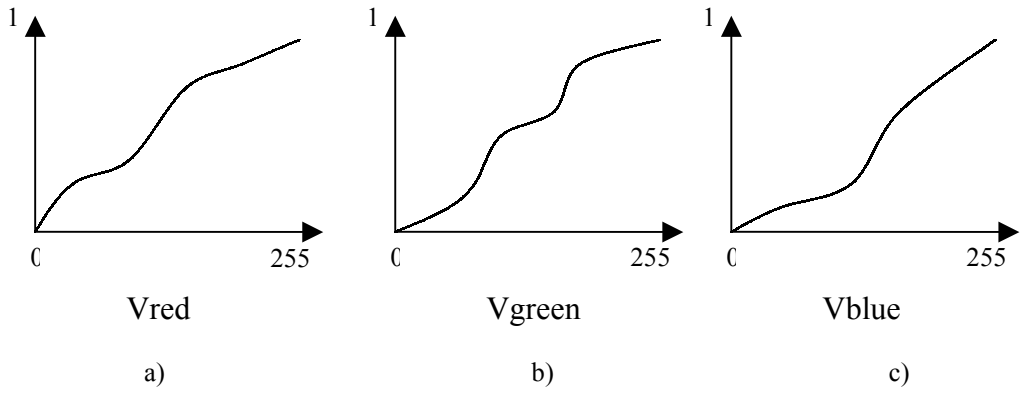


Figure 4.3.4 Examples of distribution curves. a)Red channel, b) green channel, c) blue channel.

The resulting vectors have the most important values for each colour channel. The process is similar to an intersection, speaking of set theory, because the insignificant values are not considered. This is a recursive method and it runs until the image is completely covered. After finishing the calculations with the distribution functions, follows the selection of colour shades to generate the image. Algorithm 4.3.6 shows this process.

Algorithm 4.3.6 Colour selection.

```
triple[red]=0;triple[green]=0;triple[blue]=0;
r =Math.random();flag=true;
For i=0 to 255 and flag =true
 if(Vred[i]>=r)
  triple[red]=i;flag=false;
r =Math.random();flag =true;
For i=0 to 256 and flag =true
 if(Vgreen[i]>=r)
  triple[green]=i; flag=false;
r =Math.random();flag =true;
For i=0 to 256 and flag =true
 if(Vblue[i]>=r)
  triple[blue]=i; flag=false;
```

A random number is generated and then compared with the values of the vector “Vred”. When the value of “Vred” is bigger than the random number, the loop stops (flag=false) and the index of “Vred (i)” is stored in the array “triple” in the first position. The index “i” represents the value of the red shade. The process does the same with “Vgreen” and “Vblue” and their indices are saved in the second and third position of the array “triple”. The last procedure is called “fill”, responsible of creating the image, shown in algorithm 4.3.7.

Algorithm 4.3.7 Filling the image.

```
For x=horiz to imagewidth-horiz - 1
 For y=verti to imageheight-verti - 1
 {
  construct(x,y);
  newimg2d[x][y]= triple[red] + triple[green] + triple[blue];
 }
```

The loop leaves a margin of the original image and fills the rest using the information obtained in previous procedures. The margin width is given by the values “horiz” and “verti”. It calls the construct method (algorithms 4.3.5 to 4.3.7), and obtains the shade values for each channel. The red, green and blue shades are mixed to generate the desired colour and fill the pixel at the position “(x,y)”. This method gives the new generated image.

4.3.1.3 Results

The following images are different samples obtained with the vector quantization learning method. The order of complexity is the same as the one used in the co-occurrence matrices analysis. The first images to evaluate are monochromatic and basic textures and the complexity increases to test the accuracy of this method. Its parameters are presented in table 4.3.1.

Table 4.3.1 Parameters for the vector quantization program.

Parameter	*Description*
Size	Number of vectors needed to obtain the image statistic
Dim	Dimension (in pixels) of the vectors area
Epochs	Number of iterations necessary to train the initial vectors
Alpha	Training factor defined as "1/Epochs" (see figure 4.3.2)

4.3.1.3.1 First Category: Plain Textures

The first image to analyse is a monochromatic sample (created with Microsoft Paint 5.1 ™). Its initial parameters are size=50, dim=3, epochs=50 and alpha=1/50. A margin of 10 pixels is selected and remains constant for the simple and regular textures. Figure 4.3.5a shows the original sample, figure 4.3.5b, displays an intermediate step that lefts only the original margin and clears the centre area, which is restored by the filling method. Figure 4.3.5c is the perfectly restored sample.

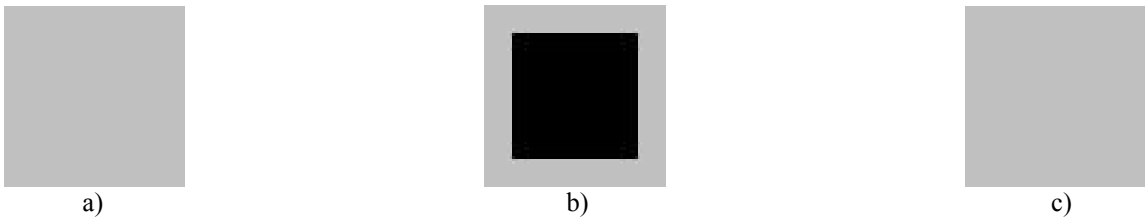


Figure 4.3.5 a) Monochromatic sample, b) intermediate step and c) its reconstruction.

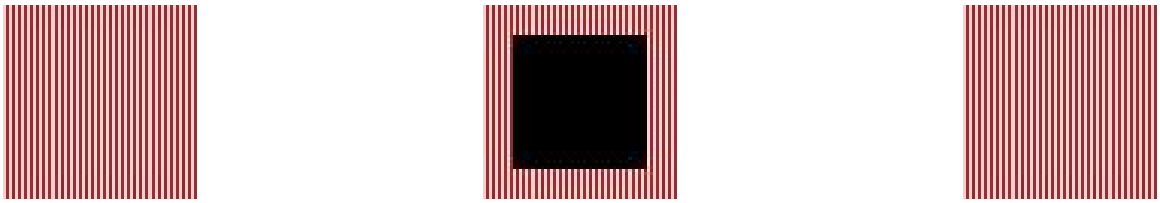
Since it is a solid texture, the parameters do not affect the reconstruction of the image. The image is flawless restored because the sample vectors can only take the value of 192 for each colour channel, so their distribution curve is a step function, see figure 4.3.6.

Figure 4.3.6 Distribution function for the monochromatic sample.

The training for these sample vectors is not necessary, since all of them contain the same value. In the reconstruction method, only one value can be chosen, 192. In conclusion, for monochromatic textures the number of vectors and iterations is irrelevant, the image is always restored as the original sample. Obviously there should be at least one vector defined, but the training can be excluded.

4.3.1.3.2 Second Category: Regular Textures

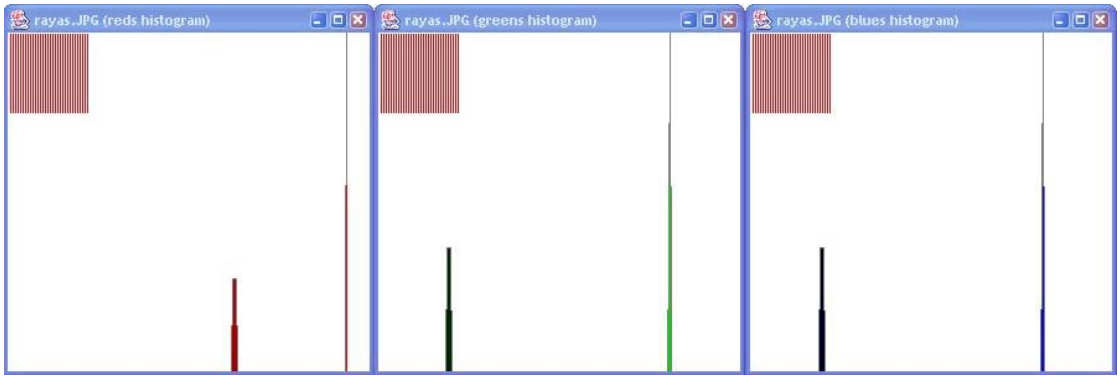
The next type of texture is regular, like the vertical pattern (created with Microsoft Paint 5.1 ™) shown in figure 4.3.7a; its parameters are size=50, dim=3, epochs=50 and alpha=1/50.



a) b) c)

Figure 4.3.7 Vertical pattern a) original sample, b) intermediate step and c) its reconstruction.

After running the program, an exact copy of the original texture is obtained, as shown in figure 4.3.7c. Since it is a regular pattern, the initial vectors are very similar between them (if not the same), although they are selected randomly through the image. The training for these vectors can be achieved in a small amount of iterations. Its distribution curves are similar to its histograms, displayed in figure 4.3.8.



a) b) c)

Figure 4.3.8 Histograms for the vertical pattern. a)red histogram, b) green histogram and c) blue histogram.

4.3.1.3.3 Third Category: Patterned Textures

The next type of textures to analyse is more complex than the regular textures; these new textures are filled with defined patterns. They remain regular, but not as homogeneous as the vertical pattern (see figure 4.3.7a). An example of these textures is shown in figure 4.3.9; it is a Saturn sample taken from "Planetary Textures for model construction" website [44].

Its colour histograms are displayed in figure 4.3.10. As observed, its spectra are completely different and its ranges vary as well. However, colour histograms are not a determinant factor for the reconstruction; they give only an idea of the distribution functions.

Figure 4.3.9 Saturn sample.

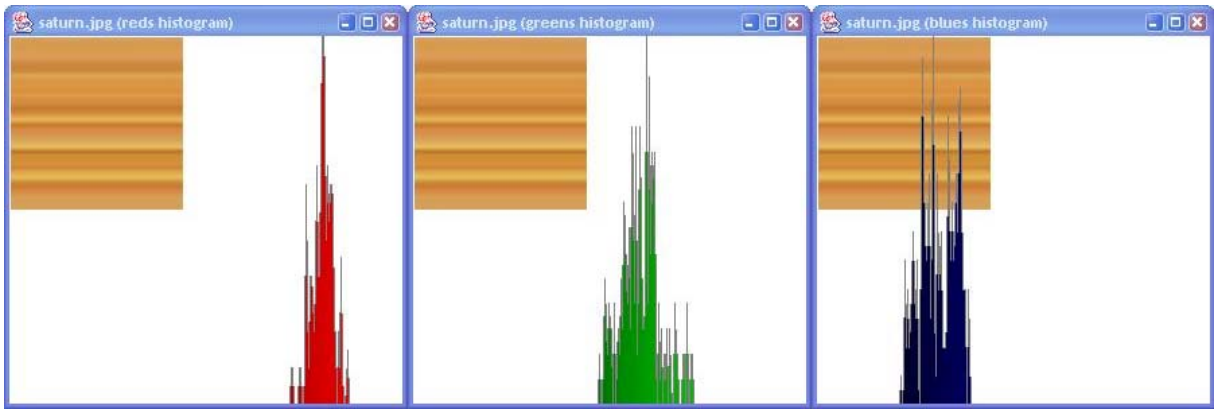


Figure 4.3.10 Histograms for Saturn sample. a)red histogram, b) green histogram and c) blue histogram.

There are three important parameters for the reconstruction; this is the reason why the number of results combination is higher than with the co-occurrence matrices method (see chap.4.2). Due to the large amount of results obtained, they are displayed in groups of three images, compared to the original sample. The parameters for the first results are displayed in table 4.3.2 and the obtained images are displayed in figure 4.3.11.

Table 4.3.2 Parameters for Saturn sample reconstruction of figure 4.3.11.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

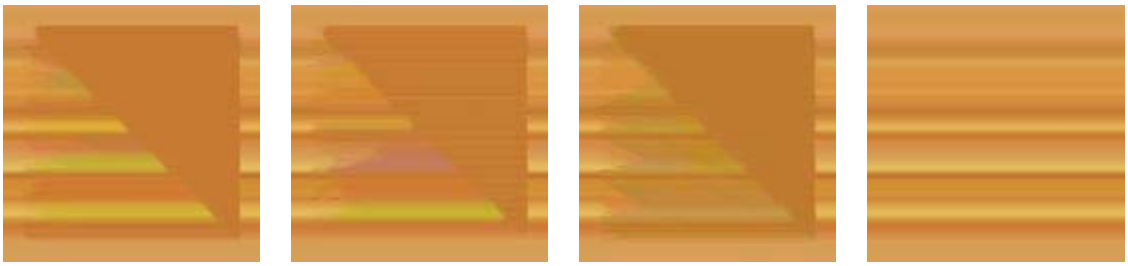
Figure 4.3.11 shows the first group of results for the Saturn texture , a)shows the result with 50 epochs, b)with 100 epochs, c)with 200 epochs and d) original sample.

As observed, the results vary depending on the iteration loops, not only for the loops of training, but also for the variable alpha (1/epochs). This group of samples has 50 training vectors with an area of 3x3 pixels each. The first image is generated with 50 iterations and it can be seen a trace of the original pattern. The second image is obtained after 100 iterations and loses the pattern; only in the bottom part of the image can be seen. The last image of this block is generated in 200 iterations and nears to the original sample.

The next set of results is obtained with different vector area and a higher number of vectors, the parameters are displayed in following table. Figure 4.3.12 shows the obtained results.

Table 4.3.3 Parameters for Saturn sample reconstruction of figure 4.3.12.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



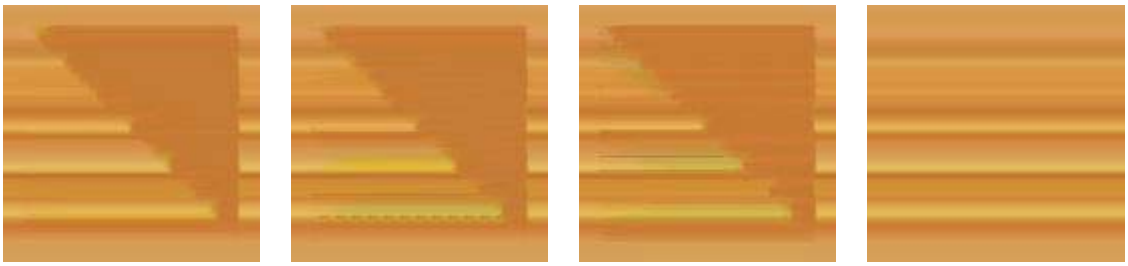
a) b) c) d)

Figure 4.3.12 shows the second group of results for the Saturn texture , a)shows the result with 50 epochs, b)with 100 epochs, c)with 200 epochs and d) original sample.

These results fill also half of the image with a solid colour, but the image obtained in 50 epochs looks similar to the original sample. The second and third images have less similarity to the Saturn texture, but they also trace recognizable patterns in the first half of the image. The next tests run with different working area and even higher amount of vectors; table 4.3.4 displays the parameters for the following tests.

Table 4.3.4 Parameters for Saturn sample reconstruction of figure 4.3.13.

Parameter	*Description*	*Value*
Size	Number of vectors.	200
Dim	Dimension of the vectors area.	7x7 pixels
Epochs	Number of iterations.	50,100 and 200
Alpha	Training factor.	1/50, 1/100 and 1/200



a) b) c) d)

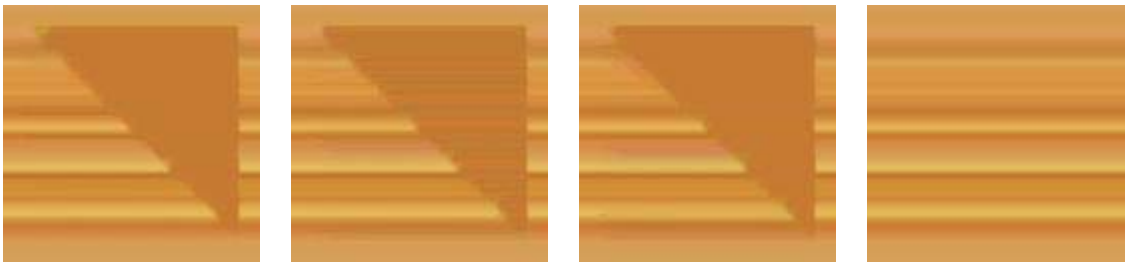
Figure 4.3.13 shows the third group of results for the Saturn texture , a)shows the result with 50 epochs, b)with 100 epochs, c)with 200 epochs and d) original sample.

The left image of this set runs with 50 epochs and its pattern has almost the same texture as the original image, but only the first half. The second result is obtained in 100 epochs and the quality decreases, especially at the bottom. The third result decreases in quality but traces a slight pattern in the other half of the image; it is not solid coloured anymore.

After evaluating the previous results, a 5x5 vectors area is selected; this area is large enough to acquire the required information and also small enough not to cause a data overflow. The number of vectors is increased to 300. Table 4.3.5 shows the rest of parameters. The obtained images are displayed in figure 4.3.14.

Table 4.3.5 Parameters for Saturn sample reconstruction of figure 4.3.14.

Parameter	*Description*	*Value*
Size	Number of vectors.	300
Dim	Dimension of the vectors area.	5x5 pixels
Epochs	Number of iterations.	50,100 and 200
Alpha	Training factor.	1/50, 1/100 and 1/200



a) b) c) d)

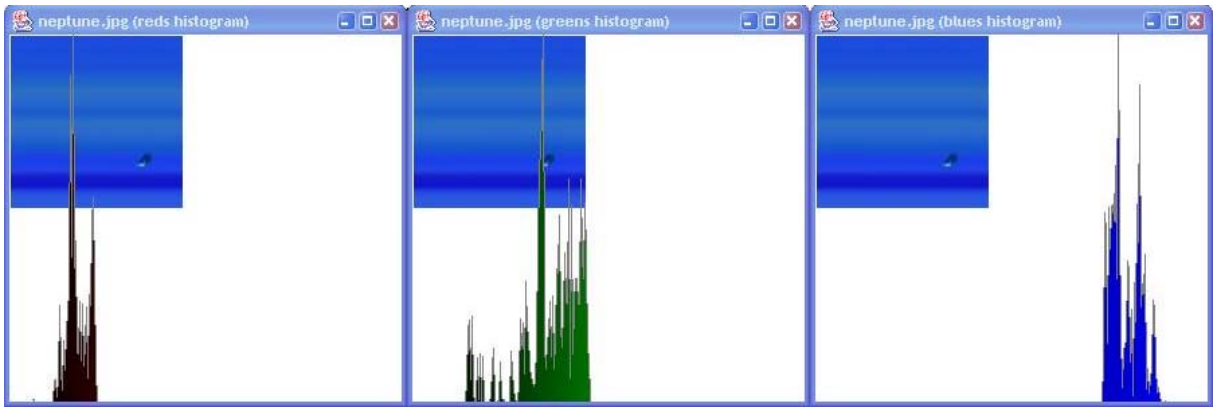
Figure 4.3.14 shows the fourth group of results for the Saturn texture, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first image (50 epochs) has almost a perfect copy of the Saturn pattern, but the second section is filled with a solid colour. The second result (100 epochs) traces a better pattern for the second half, but it differs greatly from the first section of the same image. The third one (200 epochs) traces slight patterns in the second half of the image, but the pattern in the first section looses some quality in the upper part.

The following texture is a Neptune sample, shown in figure 4.3.15.

Figure 4.3.15 Neptune surface sample.

The colour distribution for this sample is given by its histograms, see figure 4.3.16.



a) b) c)

Figure 4.3.16 Histograms for Neptune sample. a)red histogram, b) green histogram and c) blue histogram.

Its red and blue histograms have narrow range, the green one is the wider and more complicated. Due to the high amount of results, they are displayed in groups of three images, compared to the original texture. The initial parameters for this sample are located in table 4.3.6 and its results, displayed in figure 4.3.17.

Table 4.3.6 Parameters for Neptune sample reconstruction of figure 4.3.17.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

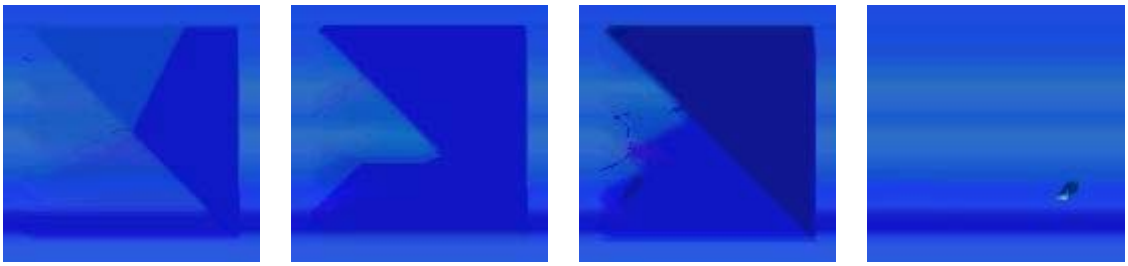
a) b) c) d)

Figure 4.3.17 shows the first group of results for the Neptune sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

As observed, the first result, figure 4.3.17a, starts to trace the original pattern, but fades away. The second and third results trace only half of the Neptune sample. The first image is the most similar to the original sample, its colour distribution and stripes approximate to the original sample. The next experiment is done with 100 vectors with wider area than the previous. Table 4.3.7 shows the parameters for this test and figure 4.3.18 displays the obtained results.

Table 4.3.7 Parameters for Neptune sample reconstruction of figure 4.3.18.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

Figure 4.3.18 shows the second group of results for the Neptune sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first image, figure 4.3.18a, throws the best result of this set of experiments. The first half is very similar to the original sample, while the second and third, start to trace it but fill the second half with a solid colour. The next experiments are calculated with 200 vectors and even wider area. Table 4.3.8 shows the parameters. The results for this test are displayed in figure 4.3.19.

Table 4.3.8 Parameters for Neptune sample reconstruction of figure 4.3.19.

Parameter	*Description*	*Value*
Size	Number of vectors	200
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.19 shows the third group of results for the Neptune sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

In this set of results, all of them approach greatly to the original texture. They have almost the same colour distribution, but only the first half. The second half still has one solid colour filling. The last experiments for this sample in this section are calculated with 300 vectors and a working area of 5x5 pixels. The parameters are shown in table 4.3.9. Figure 4.3.20 displays the results obtained for these parameters.

Table 4.3.9 Parameters for Neptune sample reconstruction of figure 4.3.20.

Parameter	*Description*	*Value*
Size	Number of vectors	300
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

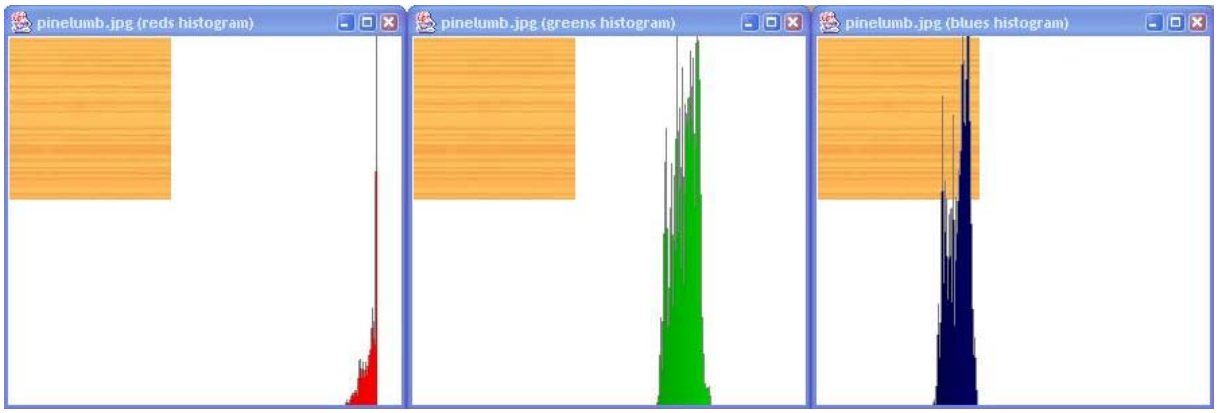
Figure 4.3.20 shows the fourth group of results for the Neptune sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The second result, figure 4.3.20b, is almost an exact copy of the original sample, even the slight colour variations are represented, but only in the first section. The first and third results show also good approximations to the original sample, but not as accurate as the result obtained with 100 epochs.

The next sample to analyse is a wooden texture, taken from Microsoft PowerPoint 2000 ™ clipart gallery. Figure 4.3.21 shows this sample.

Figure 4.3.21 "Pinelumb" sample.

This "pinelumb" sample is a regular pattern texture with slight imperfections along the stripes. However, the stripes are well defined and the contrast between them is clear, what makes the colour categorization easier. The colour distribution for this sample is represented by its histograms in figure 4.3.22. As observed, the red values majority is located in the bright side of the spectrum, while green and blue channel have a wider range and their distribution is not so uniform.



a) b) c)

Figure 4.3.22 Histograms for the "pinelumb" sample. a)red histogram, b) green histogram and c) blue histogram.

The first test for this sample is calculated with 50 vectors and a working area of 3x3 pixels. Table 4.3.10 shows the parameters for this experiment, whose results are displayed in figure 4.3.23.

Table 4.3.10 Parameters for the "pinelumb" sample reconstruction of figure 4.3.23.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.23 shows the first group of results for the "pinelumb" sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first result obtained, figure 4.3.23a, is a very similar approach to the original sample, the stripes arrangement differs from the margin, but the colour distribution along the image lies near to the original sample. The second half of this result (50 epochs) has a darker filling, but the stripes are also recognizable. The second and third results quality is lower than the first image of this set. The next experiment is calculated with a higher number of vectors; its parameters are displayed in table 4.3.11 and the results are in figure 4.3.24.

Table 4.3.11 Parameters for the "pinelumb" sample reconstruction of figure 4.3.24.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.24 shows the second group of results for the "pinelumb" sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The result obtained in 50 epochs (fig.4.3.24a) presents almost no differences to the left margin of the image; however, the second half of the image differs from the original texture.

The result obtained after 100 epochs (fig.4.3.24b) nears to the original sample in both sections of the image, but it is still recognizable from the original "pinelumb". The third result is the less accurate from this set, but still displays similarities with the left margin. The following experiments are calculated with 200 vectors. Table 4.3.12 shows the parameters for this test; figure 4.3.25 displays the obtained results.

Table 4.3.12 Parameters for the "pinelumb" sample reconstruction of figure 4.3.25.

Parameter	*Description*	*Value*
Size	Number of vectors	200
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.25 shows the third group of results for the "pinelumb" sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

From this set of results, the second experiment (fig.4.3.25b) is the most accurate reconstruction. It traces the pattern along the whole image, not only one half, and the colour distribution remains similar to the original sample. The results obtained in 50 and 200 epochs trace also the wooden pattern, but this pattern fades away in the second half of the image. The next experiments are calculated with 300 vectors. The rest of parameters are shown in table 4.3.13; figure 4.3.26 shows the obtained results.

Table 4.3.13 Parameters for the "pinelumb" sample reconstruction of figure 4.3.26.

Parameter	*Description*	*Value*
Size	Number of vectors	300
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.26 shows the fourth group of results for the "pinelumb" sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The accuracy displayed in the first half of these images is higher than the previous experiments. There are almost no differences between the margin and the reconstructed section, which indicates the importance of having 300 initial vectors for the calculation. However, the quality in the second half diminishes and there are almost no traces of stripes.

The following texture to analyse is the brick sample shown in figure 4.3.27. This image is more complex than the previous because of the joints between the bricks. It has also slight irregularities in the bricks; they are darker in some spots. The colour distribution gives a better idea of the image complexity. Its histograms (see figure 4.3.28) show a very similar range for each channel. The spectra are also quite similar to each other. The first experiments for this sample are obtained with the parameters shown in table 4.3.14; Figure 4.3.29 shows these results.

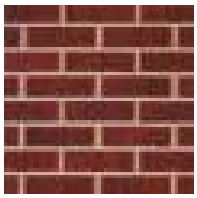
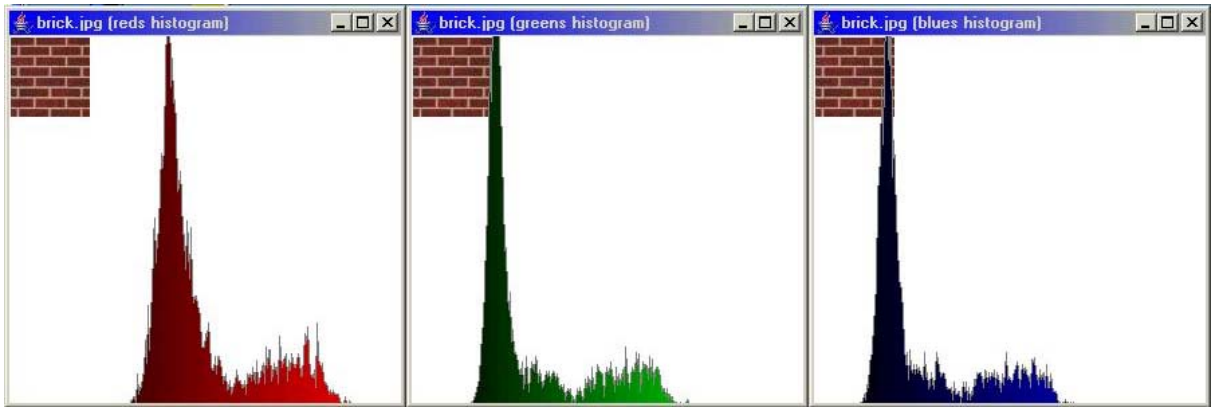


Figure 4.3.27 Brick sample.

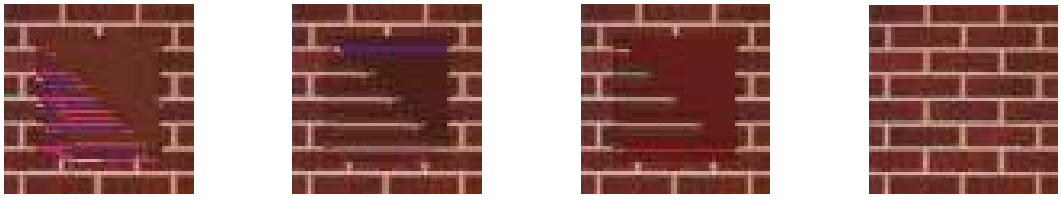


a) b) c)

Figure 4.3.28 Histograms for the bricks sample. a)red histogram, b) green histogram and c) blue histogram.

Table 4.3.14 Parameters for the brick sample reconstruction of figure 4.3.29.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



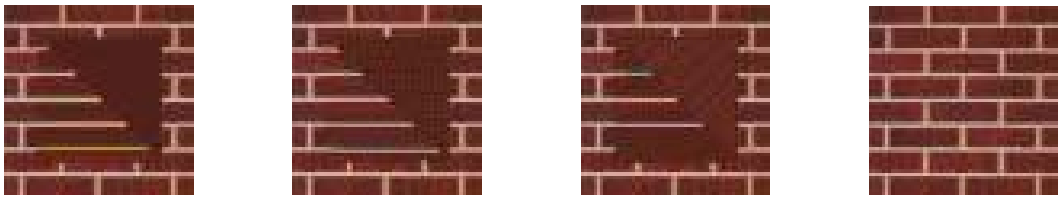
a) b) c) d)

Figure 4.3.29 shows the first group of results for the brick sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

From the first three results, the one obtained in 100 epochs (fig.4.3.29b) gets nearer to the original sample. Although it does not trace the intermediate joints, it respects the colour distribution and the stripes order. The first image (50 epochs) has almost no similarity with the original pattern. The third image (200 epochs) traces also the stripes from the left margin but there are some colour shades that do not correspond to the original sample. The next experiments parameters are shown in table 4.3.15 and the obtained results are displayed in figure 4.3.30.

Table 4.3.15 Parameters for the brick sample reconstruction of figure 4.3.30.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



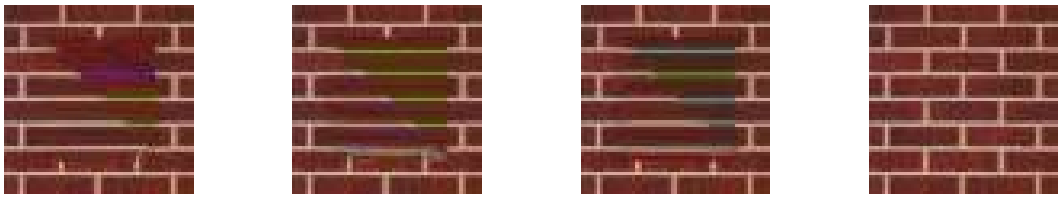
a) b) c) d)

Figure 4.3.30 shows the second group of results for the brick sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The results in this second group approach even more to the brick sample. The first and second images (fig.4.3.30a, fig.4.3.30b) display almost a copy of the original stripes, but only the first half of the image. The third image (200 epochs) traces also the joints of the bricks horizontally, but not as accurate as the previous results. The following experiments for this sample are calculated with the parameters of table 4.3.16. Figure 4.3.31 shows the results obtained.

Table 4.3.16 Parameters for the brick sample reconstruction of figure 4.3.31.

Parameter	*Description*	*Value*
Size	Number of vectors	200
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

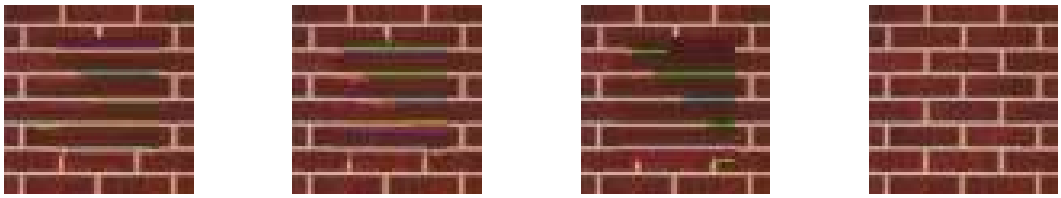
Figure 4.3.31 shows the third group of results for the brick sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

This set of images starts to resemble the original image in both sections of the image. The first result (fig.4.3.31a) leaves only a small section with a solid colour filling.

The second and third images trace the bricks and joints along the whole image. However, there are slight colour variations in the joints. The next experiments for this brick sample are calculated with the parameters of table 4.3.17. The obtained results are displayed in figure 4.3.32.

Table 4.3.17 Parameters for the brick sample reconstruction of figure 4.3.32.

Parameter	*Description*	*Value*
Size	Number of vectors	300
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

Figure 4.3.32 shows the fourth group of results for the brick sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first image (fig.4.3.32a) is a very approximate result to the original brick sample. It has slight green shades in the joints, but the brick colour is almost the same as in the original texture. The second and third results (fig.4.3.32b, fig.4.3.32c) show more irregularities in the colour shade of the joints, but they still trace an approximate pattern of the original sample.

4.3.1.3.4 Fourth Category: Irregular Patterned Textures

The next type of image to analyse is more complex than the previous analysed textures. It is a sample of Jupiter surface (see figure 4.3.33).

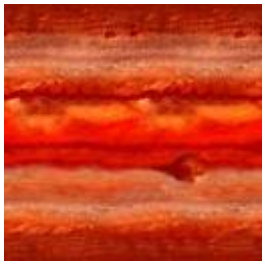


Figure 4.3.33 Jupiter sample.

This sample has clearly more irregularities than the previous textures. Its stripes are not regular anymore and contain many different shades and some darker spots. Figure 4.3.34 shows the histograms for this sample and gives a better idea of its colour distribution.

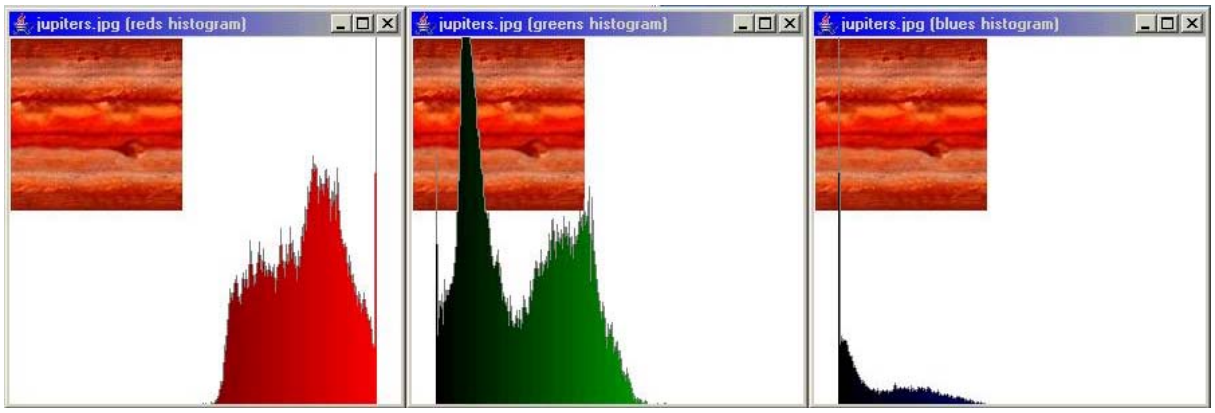


Figure 4.3.34 Histograms for Jupiter sample. a)red histogram, b) green histogram and c) blue histogram.

Each channel presents irregular spectrum with many maximum values. The range is also very different for each colour. The margin left to reconstruct this image is set to 35 pixels. Table 4.3.18 shows the rest of parameters for the first group of calculations of this sample. Figure 4.3.35 displays the obtained results.

Table 4.3.18 Parameters for Jupiter sample reconstruction of figure 4.3.35.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

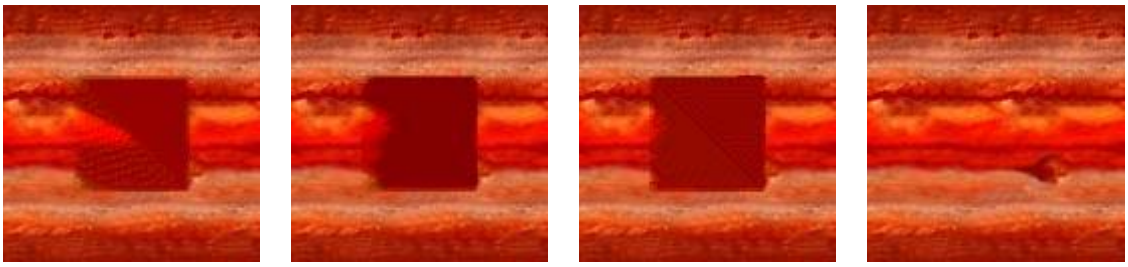
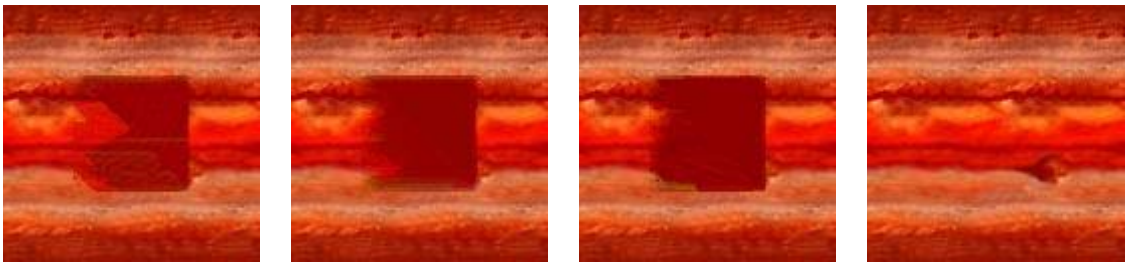


Figure 4.3.35 shows the first group of results for Jupiter sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The image has so many irregularities that the vector quantization is able to define only one solid colour to the filling. The three images have a slight pattern traced near the left margin, but it fades away. The next experiments for this sample are calculated with different parameters, the number of vectors is increased, as well as the working area. Table 4.3.19 shows the parameters for these experiments. Figure 4.3.36 shows the obtained results.

Table 4.3.19 Parameters for Jupiter sample reconstruction of figure 4.3.36.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



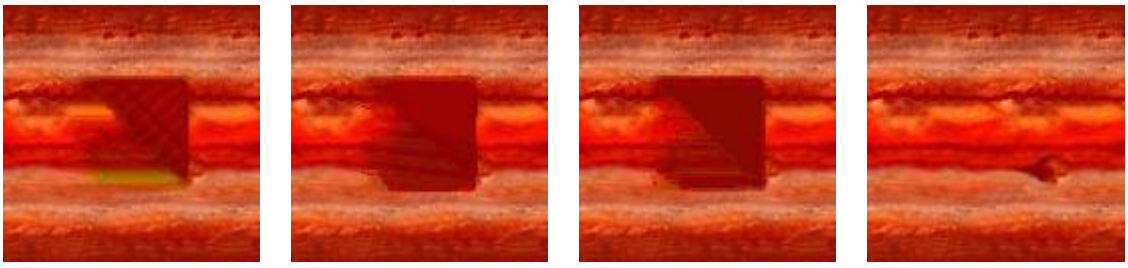
a) b) c) d)

Figure 4.3.36 shows the second group of results for Jupiter sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first image (50 epochs) starts to trace horizontal stripes, although the colour shades are different from the original sample. The second image, obtained in 100 epochs pulls the red shades from the left margin, but they fade away and are replaced with solid red colour filling. The third image (200 epochs) traces a slight pattern, but fades away. Table 4.3.20 shows the parameters for the next experiments. The obtained results are displayed in figure 4.3.37.

Table 4.3.20 Parameters for Jupiter sample reconstruction of figure 4.3.37.

Parameter	*Description*	*Value*
Size	Number of vectors	200
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



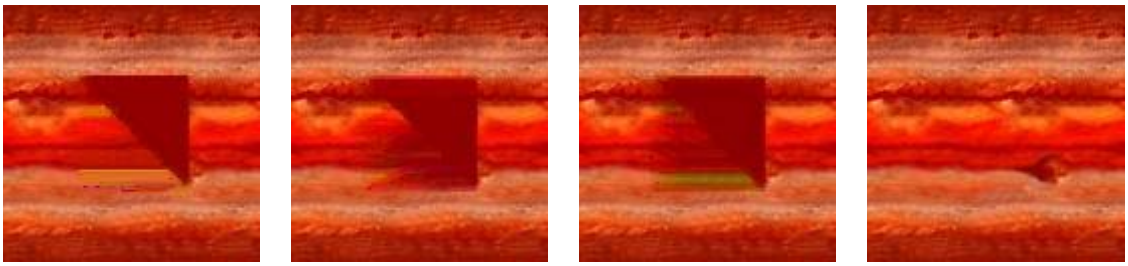
a) b) c) d)

Figure 4.3.37 shows the third group of results for Jupiter sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

From this group of pictures, the one obtained in 50 epochs (fig.4.3.37a) is the most accurate. The stripes are still not identical to the original sample, but they look alike. The second image (100 epochs) traces also the pattern from the left margin, but not until the end of the picture and fades away. The third result (200 epochs) traces also the pattern from the left margin, but only one half of the area to restore, the rest is filled with a solid colour. The next experiments are calculated with a higher number of vectors, 300. The parameters are located in table 4.3.21. Figure 4.3.38 shows the results obtained with these parameters.

Table 4.3.21 Parameters for Jupiter sample reconstruction of figure 4.3.38.

Parameter	*Description*	*Value*
Size	Number of vectors	300
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

Figure 4.3.38 shows the fourth group of results for Jupiter sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The result obtained in 50 epochs (fig.4.3.38a) displays a very similar pattern of Jupiter surface, but only in the first half of the area to restore. The colour distribution is not exactly the same, but approximates to the original sample. The second result (100 epochs) traces a different pattern, but its colour shades are not so distant from the original shades. The last result, obtained in 200 epochs, traces also a similar pattern, but the colour shades differ from the original texture.

In the next section (4.3.2), there are two important differences in the restoration process. The calculation of distance between vectors is not Euclidean anymore; it is a distance based in weights. The nearer is the pixel to the centre of vector, the more important is for the distance calculation. The working area, used in the reconstruction process, changes also from the one used in this section.

4.3.2 Second approach: Vector Learning with Weighted Distance

4.3.2.1 Obtaining Information

In previous section (4.3.1), the learning and comparing distance for vector training is the Euclidean distance. For this approach, another type of distance is implemented when training the vectors. This distance is a weighted distance, explained in more detail further in this section.

4.3.2.1.1 Defining the Initial Vectors

The initial vectors are defined as in previous section. These vectors are the ones that get the information of the image to create statistics for the image reconstruction. They are defined in a four-dimensional array as follows.

float supvector[][][][]=new float [colour][size][dim][dim];

Their indices are colour channel (red, green and blue), number of vectors, given by the constant “size” and their last two indices correspond to the vector area or quadratic dimension, defined by the constant “dim”. The initial vectors are chosen stochastically. A quadratic sample is taken from the image and its colour components are extracted and stored in the correspondent vectors. This procedure is displayed in algorithm 4.3.1.

The “size” loop defines the number of learning vectors and the coordinates (“randhor”, “randver”) set the position of the initial pixel of the sample. A square is traced from that position to the coordinates (“randhor+dim-1”, “randver+dim-1”) in the second loop. Figure 4.3.39 shows this procedure.

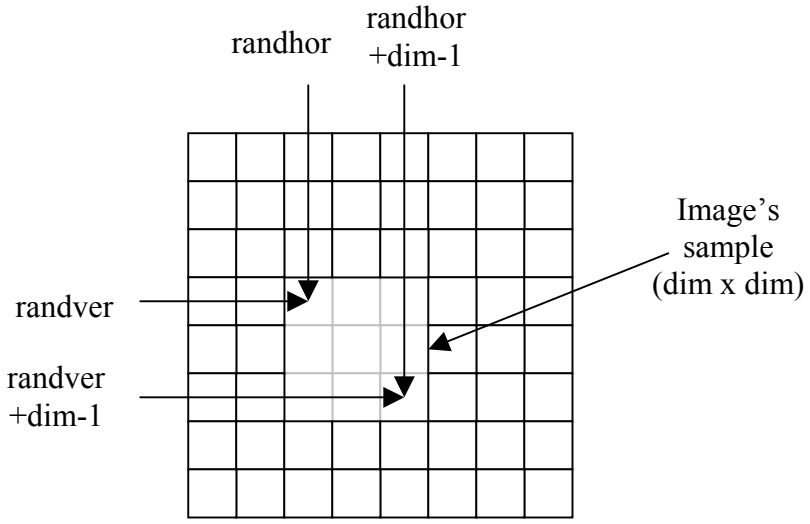


Figure 4.3.39 Random sample to store as initial vector.

The obtained sample is stored in the array "morig2d". The next loop takes the red, green and blue components from the sample and stores them in the vector array "supvector".

Distribution curve "Vecdist" represents the statistics of the colour shades of the initial vectors. The first index stores the colour component, the second index corresponds to the number of initial vectors and the last index represents the range of shades for each colour.

Distribution functions are similar to histograms; it shows the number of times each colour shade occurs when acquiring the vectors. It takes the middle value of each initial vector, extracts the colour component and stores it in the array. After the storing process, the learning process begins.

4.3.2.1.2 Vector Training

One comparative vector of the original sample is taken; its colour components are extracted and compared with each learning vector. The nearest learning vector to the comparative vector is called winner and it approaches to the comparative vector. The training method is called "winner takes all", because only the winner vector is trained by each epoch [50]. The process runs until it reaches the specified amount of epochs.

The first step for the training calculates the distance between single pixels and then, the sum of them will give the distance between the comparative vector and the learning vector. The pixel distance takes the middle pixel as the most important and the relevancy of the pixels decreases when getting further from the centre. This procedure is shown in algorithm 4.3.8.

Algorithm 4.3.8 Weighted distance calculation.

```
For w=0 to epochs
{
 takesample();
 For c=0 to size
  For i=0 to dim
   For j=0 to dim
{
```

```
    zr=supvector[red][c][i][j]-veccompar[red][i][j];
    zr*=1.0/(1.0+(i-dist)*(i-dist)+(j-dist)*(j-dist));
    distauxr+=zr*zr;
    zg=supvector[green][c][i][j]-veccompar[green][i][j];
    zg*=1.0/(1.0+(i-dist)*(i-dist)+(j-dist)*(j-dist));
    distauxg+=zg*zg;
    zb=supvector[blue][c][i][j]-veccompar[blue][i][j];
    zb*=1.0/(1.0+(i-dist)*(i-dist)+(j-dist)*(j-dist));
    distauxb+=zb*zb;
   }
}
```

Variable "zr" takes the distance between the red components of the learning vector and comparative vector, divides it by a defined factor, and squares it. This process runs for each pixel of the sample and the sum of them gives the red distance "distauxr". The same algorithm is used for the green and blue values, giving as a result "distauxg" for greens and "distauxb" for blues. The relevancy of the pixels distance is given by the factor *1.0/(1.0+(i-dist)*(i-dist)+(j-dist)*(j-dist)),* which is better explained in figure 4.3.40.

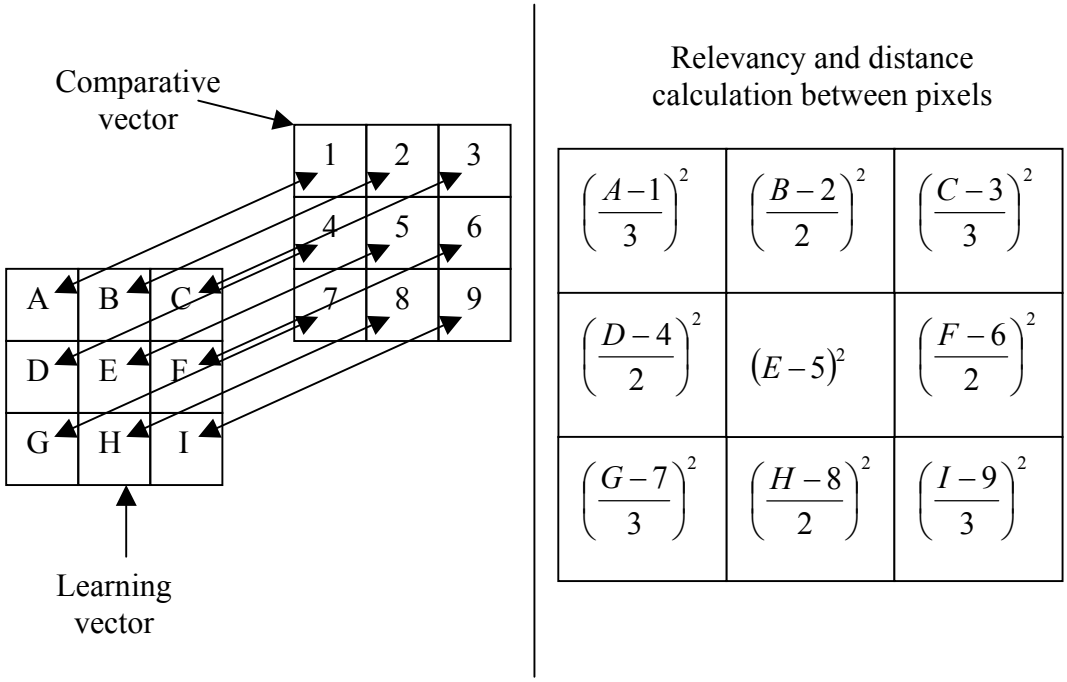


Figure 4.3.40 The central pixel relevancy is the highest; the second most important are the vertical and horizontal pixels and the least important are the diagonal pixels.

The acquired sample, defined in the "size" loop, is compared to every learning vector. After finishing the comparisons, the nearest vector to the sample is elected as the winner vector. The next step is to move the winner vector in direction of the sample, which is given by the function *winner vector = alpha * compare vector + (1-alpha) * winner vector*, where "alpha" is defined as "1/epochs".

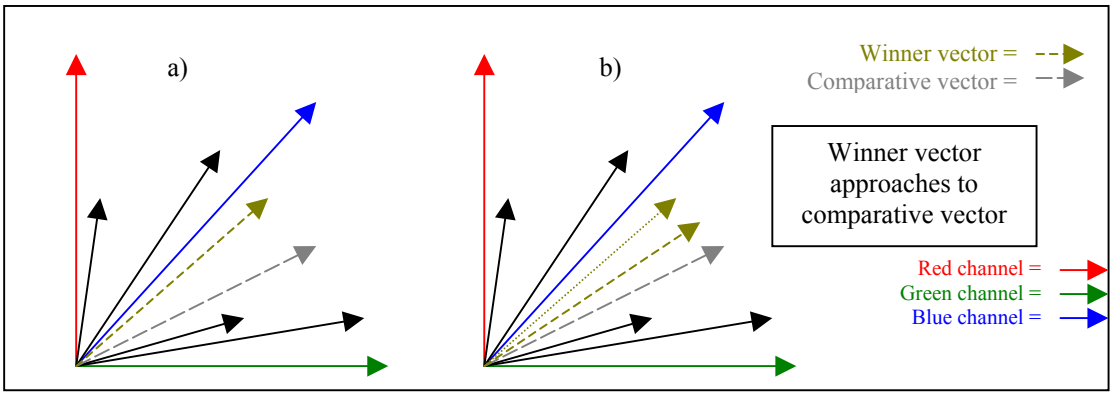


Figure 4.3.41 Vector training. a)Initial position, b)winner vector approaches to comparative vector.

The next step is the extraction of the information stored in the initial vectors. For this, the middle value of the comparative vector is taken and stored to its correspondent allocation. This method is the same as in the previous section. Algorithm 4.3.3 shows this process.

Algorithm 4.3.3 shows only the first epoch. The following epoch takes another random sample, compares it with the stored vectors, elects the winner vector, moves it in direction of the recent sample and takes the colour values of the sample to store them in the distribution functions. An example of the vector training is displayed in figure 4.3.41. The whole training is defined by the "epochs" loop. This means that an "epochs-number" of winner vectors is obtained. Some of them are winners more than once and some others maybe not even once; this is determined stochastically. The next step normalises the distribution functions.

Distribution curves have for each vector a histogram-like statistic. Also for each "vector position" index, a sum of its values is made and then every value is divided by the sum; this is the normalisation process. This section describes only the acquiring of information; the next section explains the image reconstruction with the stored data.

4.3.2.2 Image Reconstruction

After the information storing process and the vector training, it follows the construction of a new image. This process reconstructs the image by taking the information of the nearest pixels (working area) and comparing its values with the vector values.

The procedure takes a margin of the original picture as reference and then from this margin the new image is built. Figure 4.3.42 displays the reconstruction process.

The image reconstruction direction is vertical and the working area is determined by the constant "dist" (see algorithm 4.3.12); in figure 4.3.42, "dist" has the value of one. Each new generated pixel gathers information from the neighbours, defined by the working area, and compares the acquired values to the ones stored in the learning vectors. The process runs until the last pixel of the image is reached. This is a brief description of the reconstruction method; more detailed information is given in following pages.

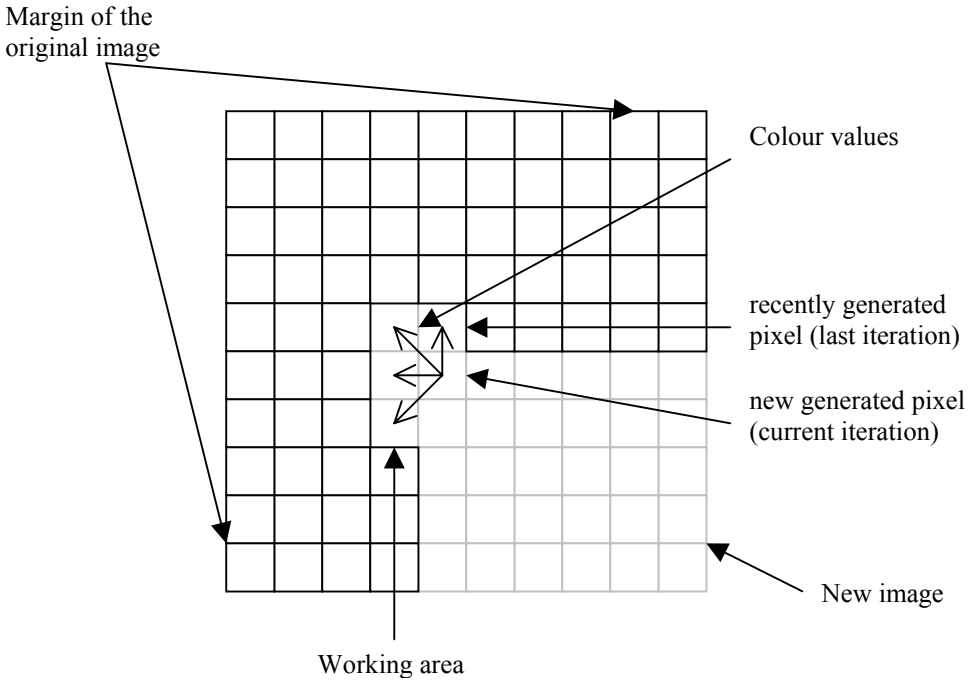


Figure 4.3.42 Reconstruction process.

The first step is the creation of three arrays, one for each colour. They store the compressed information of the textures, already acquired from the learning vectors. These arrays are 256 elements long and represent the possible values that each colour channel can choose to rebuild the image. Algorithm 4.3.9 shows this procedure.

Algorithm 4.3.9 Selection of winner vectors with weighted distance.

```
float Vred[] = new float[256];
float Vgreen[] = new float[256];
float Vblue[] = new float[256];
For c=0 to size
 For i=0 to dim
  For j=0 to dim
  {
   di=p-dist+i;dj=q-dist+j;
   if (di < p || ((di == p) && (dj < q))
||di > imagewidth-margen-1 || dj > imageheight-margen-1)
   {
    zr=supvector[red][c][i][j]-red of newimg2d[di][dj]);
    zr*=1.0/(1.0+(i-dist)*(i-dist)+(j-dist)*(j-dist));
    distauxr+=zr*zr;
    zg=supvector[green][c][i][j]-green of newimg2d[di][dj]);
    zg*=1.0/(1.0+(i-dist)*(i-dist)+(j-dist)*(j-dist));
    distauxg+=zg*zg;
    zb=supvector[blue][c][i][j]-blue of newimg2d[di][dj]);
    zb*=1.0/(1.0+(i-dist)*(i-dist)+(j-dist)*(j-dist));
    distauxb+=zb*zb;
   }
  }
```

The main loop corresponds to the constant “size”, this is, the quantity of learning vectors. The inner loop (“dim” x “dim”) defines the working area with help of the “if” condition, which selects the pixels that are taken to compare with the stored values, as shown the figure 4.3.42. In this section, the condition is changed to take only the upper and left section of the working area from the pixel position for comparison; the upper right pixel is discarded, as shown in figure 4.3.42.

Variables “p” and “q” represent the actual position of the pixel in the new image. “Zr”, “zg” and “zb” are auxiliary variables, which help to determine the distance between the vectors and the working area. The winner vector is the one with the smallest distance towards the working area.

This procedure works as in the learning stage; the difference is that the learning vectors “supvector[colour][c][i][j]” are not compared to a random sample, but to the working area, represented by “newimg2d[di][dj]”. The procedure seeks for a winner vector from which the distribution function is calculated to reconstruct the pixel at the current position. This procedure is displayed in algorithm 4.3.5. It is the same procedure used in section 4.3.1.

Since the three channels are independent, they are chosen according to the winner vectors of each colour. The following step makes the vectors cumulative and normalise them. An example of distribution functions is displayed in figure 4.3.43.

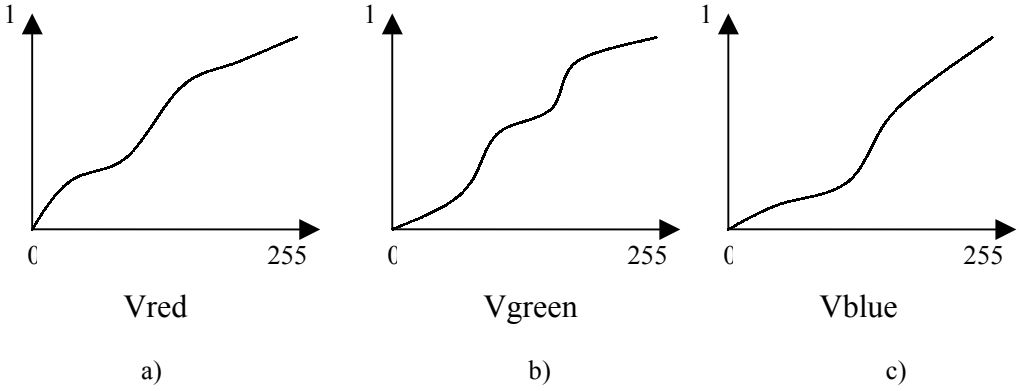


Figure 4.3.43 Examples of normalised and cumulative distribution functions.

The resulting vectors have the most important information for each colour channel. This is a recursive method and runs until the image is completely covered. After finishing the distribution functions, the colour shades for the new image are selected. The selection procedure remains the same as in previous section. Algorithm 4.3.6 shows this process.

As observed in algorithm 4.3.6, a random number is generated and then is compared with the values of the vector “Vred”. When the value of “Vred” is bigger than the random number, the loop stops (flag=false) and the index of “Vred (i)” is stored in the array “triple” in the first position. The index “i” represents the value of the red shade. The process calculates “Vgreen” and “Vblue” and their indices are saved in the second and third position of the array “triple”. The filling procedure is also the same used in section 4.3.1. Algorithm 4.3.7 shows this process.

The procedure leaves the margin of the original image and erases the rest of the image. It fills the image using the information obtained in previous procedures. The margin width is given by the values “horiz” and “verti”. It calls the construct method (algorithms 4.3.9, 4.3.5 and 4.3.6), and obtains the shade values for each channel. The red, green and blue shades are mixed together to generate the desired colour and fill the pixel at the position “(x,y)”. This method is in charge of creating the image.

4.3.2.3 Results

Different textures are tested with this vector quantization method and its results are shown in this section. The first images to evaluate are simple (monochromatic and basic textures) and the complexity level of the analysed images increases to test the accuracy of this method. The reconstruction parameters used for this experiment are represented in table 4.3.22.

Table 4.3.22 Parameters for the vector quantization program.

Parameter	*Description*
Size	Number of vectors needed to obtain the image statistic
Dim	Dimension (in pixels) of the vectors area
Epochs	Number of iterations necessary to train the initial vectors
Alpha	Training factor defined as “1/Epochs” (see figure 4.3.41)

4.3.2.3.1 First category: Plain Textures

The first image to reconstruct in this section is a monochromatic sample (created with Microsoft Paint 5.1 TM). Its initial parameters are size=50, dim=3, epochs=50 and alpha=1/50. The margin for simple and regular textures is set to 10 pixels wide. Figure 4.3.44, shows the original sample to the left, an intermediate step, that lefts only the original margin and clears the centre area, in the middle and to the right, the perfectly restored sample.

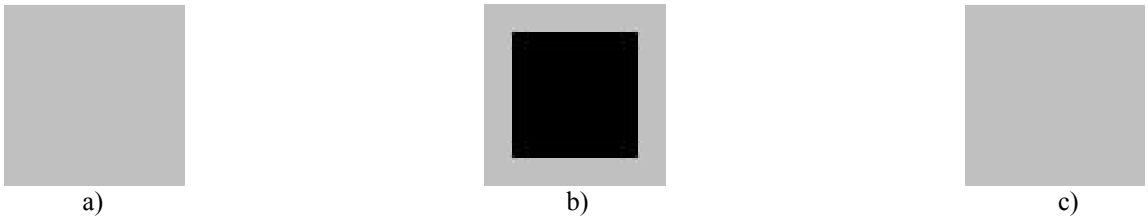


Figure 4.3.44 a) Monochromatic sample, b) intermediate step and c) its reconstruction.

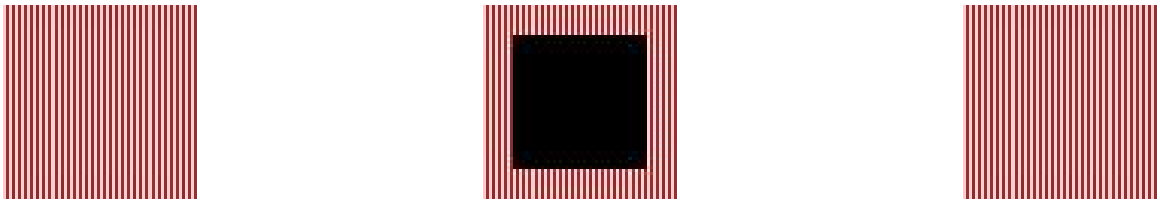
Working with solid textures is simple because the parameters are not important to the image reconstruction. Since the sample vectors can only take the value of 192 for each colour channel, their distribution curve is a step function, see figure 4.3.45, and the image is flawlessly restored.

Figure 4.3.45 Distribution function for the monochromatic sample.

Training is not necessary for these sample vectors because all of them contain the same value. At the reconstruction, they can only choose one value, 192. It is concluded that for monochromatic textures, the number of vectors and iterations is irrelevant, the image is always restored to its original state. It is clear that at least one vector must be defined, but the training can be excluded.

4.3.2.3.2 Second Category: Regular Textures

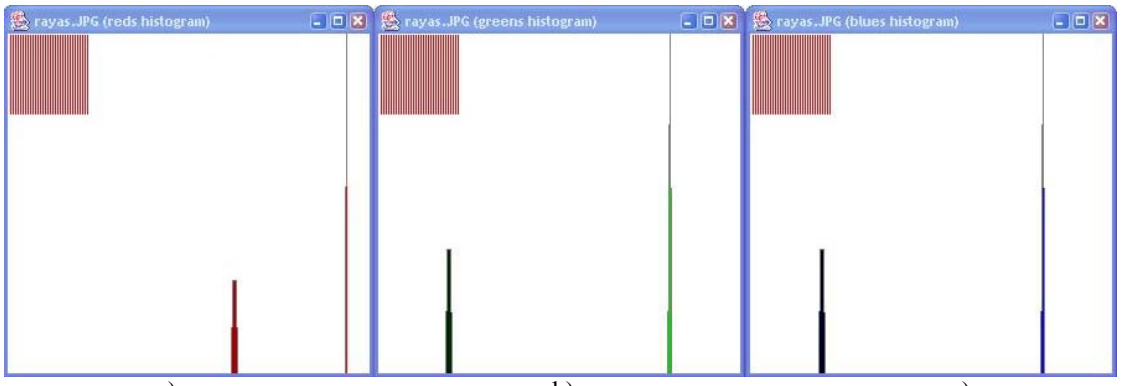
Regular textures are more complex than monochromatic textures, however, they remain as simple images. Figure 4.3.46a shows the vertical pattern to analyse (created with Microsoft Paint 5.1 ™); its initial parameters are size=50, dim=3 and epochs=50 and alpha=1/50.



a) b) c)

Figure 4.3.46 Vertical pattern a) original sample, b) intermediate step and c) its reconstruction.

After the reconstruction, the program throws a replica of the original texture, shown in figure 4.3.47c. The initial vectors are very similar between them (if not the same), although they are selected randomly through the image; this occurs due to image regularity. These vectors' training is achieved in a small amount of iterations. Its distribution curves are similar to its histograms, displayed in figure 4.3.48.



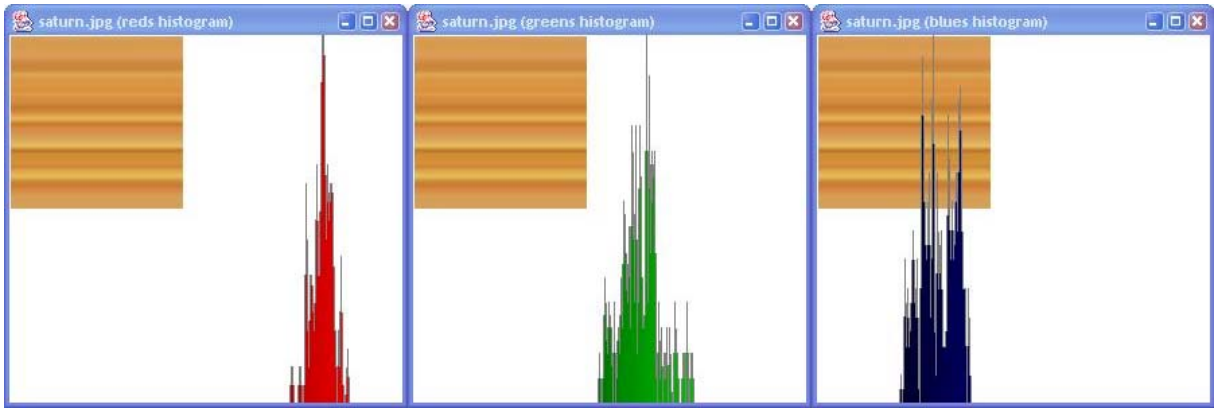
a) b) c)

Figure 4.3.48 Histograms for the vertical pattern. a)red histogram, b) green histogram and c) blue histogram.

4.3.2.3.3 Third Category: Patterned Textures

The next type of texture to analyse is more complex than the regular textures, they are filled with defined patterns. They remain regular, but not as homogeneous as the vertical pattern (see figure 4.3.46a). An example is shown in figure 4.3.49; it is a Saturn surface sample. Its colour histograms are displayed in figure 4.3.50.

Figure 4.3.49 Saturn sample.



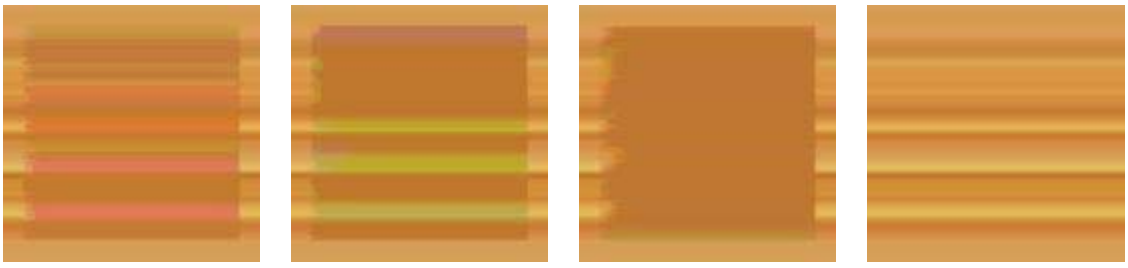
a) b) c)

Figure 4.3.50 Histograms for Saturn sample. a)red histogram, b) green histogram and c) blue histogram.

As observed in its histograms, the colours range is wider and the spectra are also more complex than the ones of the previous regular textures. The parameters for the first group of experiments are shown in table 4.3.22. The obtained images are displayed in figure 4.3.51.

Table 4.3.22 Parameters for Saturn sample reconstruction of figure 4.3.51.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

Figure 4.3.51 shows the first group of results for Saturn sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

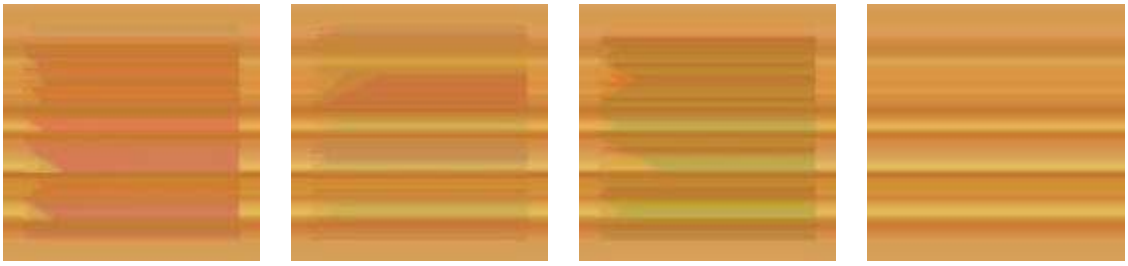
The result obtained in 50 epochs (fig.4.3.51a) traces the stripes along the entire image, not only one half like in previous section (fig.4.3.11a). However, the colour distribution is still different from the original sample. The second image (100 epochs) does not have the stripes in the upper part of the picture and the lower pattern differs from the original texture.

The third image (200 epochs) shows lower quality than the previous two. It is filled with one solid colour almost in its totality.

The next experiment increases the number of vectors; the other parameters are shown in table 4.3.23. Figure 4.3.52 displays the obtained images for this experiment.

Table 4.3.23 Parameters for Saturn sample reconstruction of figure 4.3.52.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

Figure 4.3.52 shows the second group of results for Saturn sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

As observed, in this section, the problem of filling only half of the image is already fixed. It is only matter of finding the correct combination of parameters to rebuild the sample as similar as possible to the original texture.

The first result (50 epochs), fills the image with different colour shades and the stripes are not so well defined. The second image (100 epochs) approaches to the Saturn surface texture, it still has slight green shades that do not correspond to the original sample. The third image (200 epochs) starts to loose some quality and more green shadows are present in the lower part. However, the increasing of vectors from 50 to 100 shows a major improvement in the image quality.

The next experiment runs with 200 vectors. Table 4.3.24 shows the other parameters. The obtained results for these parameters are displayed in figure 4.3.53.

Table 4.3.24 Parameters for Saturn sample reconstruction of figure 4.3.53.

Parameter	*Description*	*Value*
Size	Number of vectors	200
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

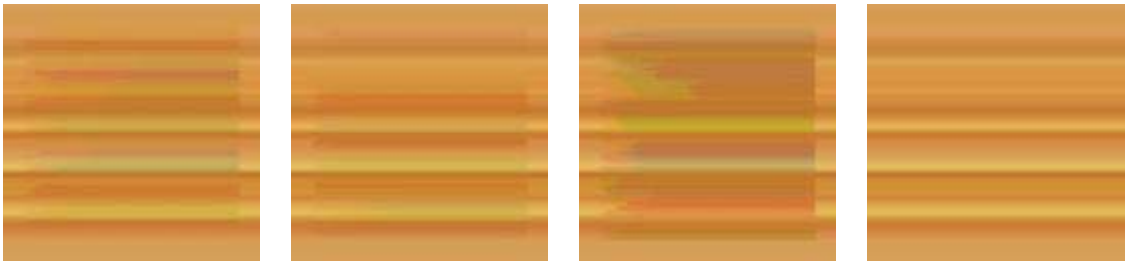
Figure 4.3.53 shows the third group of results for Saturn sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The image obtained in 50 epochs (fig.4.3.53a) is a very accurate reconstruction; there are no noticeable differences with the original texture. The colour shades are the same and the width of stripes corresponds also to the original Saturn sample. The second image (100 epochs) shows also great similarity to the original texture, but it is not as accurate as the first result. The third image quality (200 epochs) decreases; the upper part is filled with a solid colour instead of the corresponding stripes.

The last analysis of this texture is calculated with the parameters shown in table 4.3.25. The number of vectors is increased to 300 and the working area is reduced to 3x3 pixels. Figure 4.3.54 shows the results obtained with these parameters.

Table 4.3.25 Parameters for Saturn sample reconstruction of figure 4.3.54.

Parameter	*Description*	*Value*
Size	Number of vectors	300
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

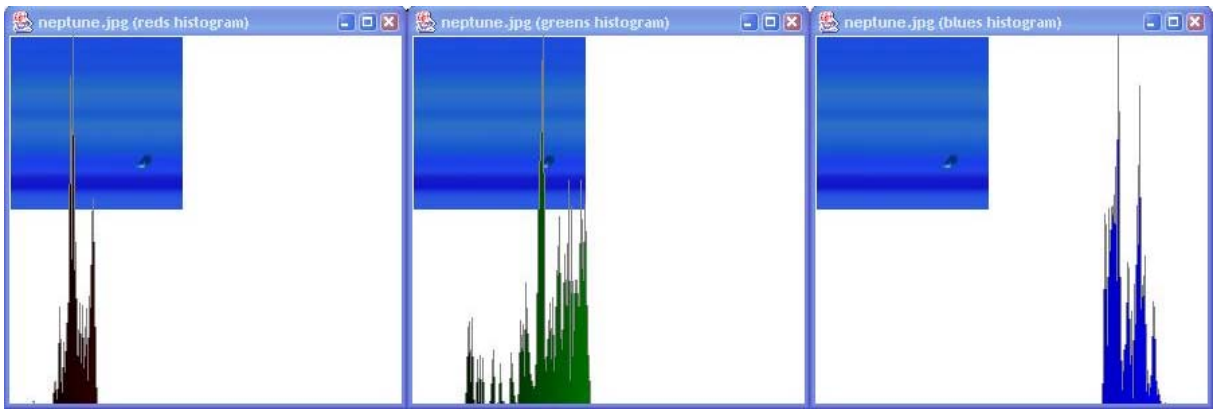
Figure 4.3.54 shows the fourth group of results for Saturn sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first result (50 epochs) displays a pattern similar to the original, but it also shows some green shadows in the middle section of the image. The result obtained in 100 epochs (fig.4.3.54b) displays a better reconstruction of this Saturn sample, but it is not as accurate as the one obtained in the previous group with 200 vectors and 50 epochs (fig.4.3.53a). The third image (200 vectors) traces also a striped pattern, but its quality is not comparable to the previous results.

The next image to analyse is a Neptune surface sample, shown in figure 4.3.55.

Figure 4.3.55 Neptune surface sample.

Its pattern is also horizontal oriented but it has a small irregularity in the lower right section. Its colour distribution is given by its histograms, shown in figure 4.3.56.



a) b) c)

Figure 4.3.56 Histograms for Neptune sample. a)red histogram, b) green histogram and c) blue histogram.

As observed, its colours spectra and ranges are more complex than the ones in Saturn or the regular textures. The first experiment for this sample runs with 50 vectors; each vector has an area of 3x3 pixels. The rest of parameters are located in table 4.3.26. The results obtained in this experiment are displayed in figure 4.3.57.

Table 4.3.26 Parameters for Neptune sample reconstruction of figure 4.3.57.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.57 shows the first group of results for Neptune sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first image (50 epochs) traces the original pattern but only in the upper and lower parts, the rest is filled with different colour shades. The result obtained in 100 epochs training (fig.4.3.57b) shows also the original pattern in the upper and lower part, and the middle is filled with one solid colour. The third image (200 epochs) is not accurate; it is filled with one solid colour and the stripes are not present.

A different analysis is calculated with a higher number of vectors. Table 4.3.27 shows the parameters for this analysis. The results obtained are displayed in figure 4.3.58.

Table 4.3.27 Parameters for Neptune sample reconstruction of figure 4.3.58.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

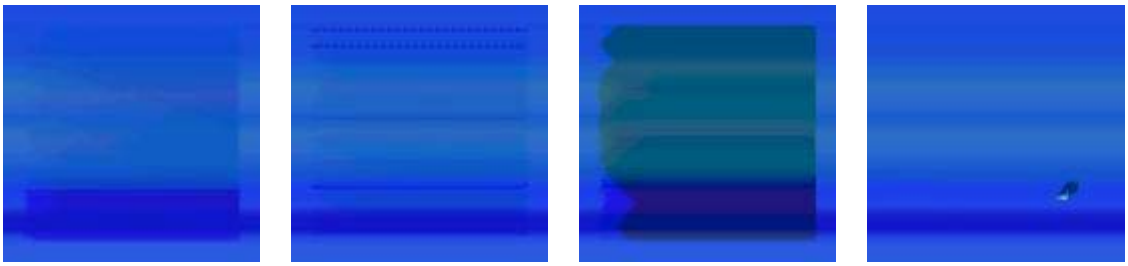
Figure 4.3.58 shows the second group of results for Neptune sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The results obtained in this group are not satisfactory either. The first image (50 epochs) displays, in the bottom and upper parts, a pattern similar to the original Neptune surface, but the middle is filled with different colour shades and the stripes do not cover the whole image. The image obtained in 100 epochs training (4.3.58b) nears to the original sample, but the middle still remains distant from the original texture. The third result (200 epochs), traces an irregular pattern along the image; its quality is lower than the previous result of this group.

The next experiment is calculated with 200 vectors. Table 4.3.28 shows the other parameters used for this test. Figure 4.3.59 displays the obtained results.

Table 4.3.28 Parameters for Neptune sample reconstruction of figure 4.3.59.

Parameter	*Description*	*Value*
Size	Number of vectors	200
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

Figure 4.3.59 shows the third group of results for Neptune sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

As observed, these results approach to the original sample, specially the first and second image. The result obtained in 50 epochs training (4.3.58a) traces a very similar pattern to the original sample. It has slight irregularities in the middle section, but is still a good reconstruction. The second image (100 epochs), displays also an accurate reconstruction, even the slight irregularity on the lower section of the original sample is covered. The third result (200 epochs) shows a low quality reconstruction, the stripes width is similar, but the colour shades are out of range.

The last experiment with this sample is done with the parameters of the table 4.3.29. The number of vectors is increased to 300 and the area remains of 7x7 pixels. Figure 4.3.60 shows the results obtained with these parameters.

Table 4.3.29 Parameters for Neptune sample reconstruction of figure 4.3.60.

Parameter	*Description*	*Value*
Size	Number of vectors	300
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

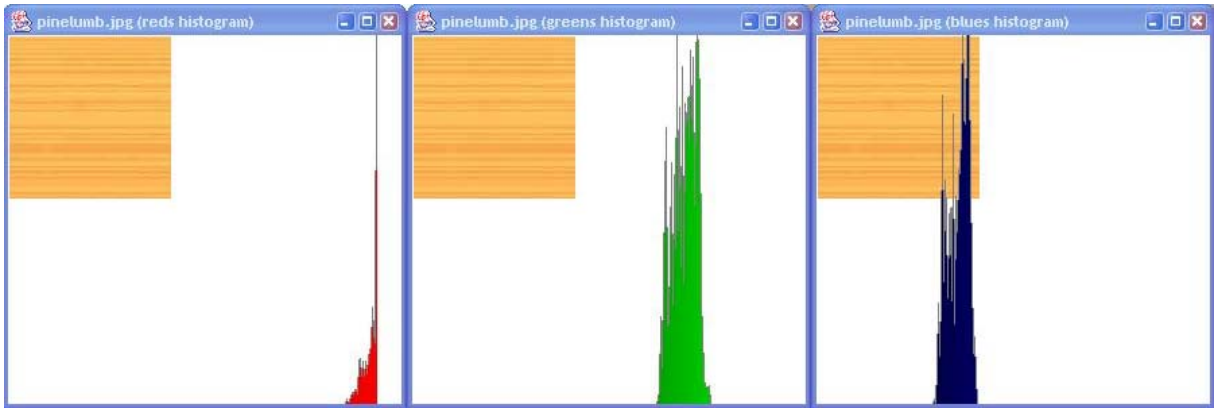
a) b) c) d)

Figure 4.3.60 shows the fourth group of results for Neptune sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

As observed, the reconstruction of this sample is also possible. The result obtained in 50 epochs training (fig.4.3.60a) is almost a flawless restoration of the original sample, even the small irregularity is covered with the surrounding pattern. The second image (100 epochs) is also a fine reconstruction of the Neptune surface sample, although its quality is slightly lower than the first image (50 epochs). The result obtained in 200 epochs training is not as accurate as the previous, the pattern is not established and the colour shades vary from the original texture.

The next sample to analyse is a wooden texture, taken from Microsoft PowerPoint 2000 ™ clipart gallery, shown in figure 4.3.59. This wooden pattern is also regular, it is horizontal oriented and its irregularities are inexpedient. Its colour distribution is shown in figure 4.3.60.

Figure 4.3.59 "Pinelumb" sample.



a) b) c)

Figure 4.3.60 Histograms for the "pinelumb" sample. a)red histogram, b) green histogram and c) blue histogram.

The histograms spectra are simpler than the previous planet surfaces samples. Its colours range is also narrower than the previous analysed textures. The first experiment runs with 50 vectors and a working area of 3x3 pixels. Table 4.3.30 shows the other parameters. The obtained results are located in figure 4.3.61.

Table 4.3.30 Parameters for the "pinelumb" sample reconstruction of figure 4.3.61.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.61 shows the first group of results for the "pinelumb" sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first result, obtained in 50 epochs training, (fig.4.3.61a) traces the stripes along the image but the colour does not match the original sample. The result obtained in 100 epochs training (fig.4.3.61b) is a very accurate reconstruction, it has imperfections that are barely noticeable, but the lines width and colour shades are almost a copy of the original. The third result (200 epochs) is not as accurate as the previous two, the stripes are not traced uniformly and the colours are also darker than the original "pinelumb" sample.

For the next experiment, the number of vectors is increased to 100. Table 4.3.31 shows the other parameters. Figure 4.3.62 displays the obtained results.

Table 4.3.31 Parameters for the "pinelumb" sample reconstruction of figure 4.3.62.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.62 shows the second group of results for the "pinelumb" sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The accuracy for this set of pictures is increased greatly. The result obtained in 50 epochs training (fig.4.3.62a) is an almost exact replica of the original texture. Its colour shades and stripes are not distinguishable from the original wooden texture. The second result (100 epochs) shows also a very good reconstruction, it has slight imperfections in the middle section, but is barely noticeable. The third image (200 epochs) displays a lower quality reconstruction, the stripes are not as uniform distributed as in the previous results and the colour shades are darker than the original.

In the next experiments, the number of vectors is increased to 200 and the area reduced to 3x3 pixels. Table 4.3.32 shows the rest of parameters; figure 4.3.63 displays the obtained results.

Table 4.3.32 Parameters for the "pinelumb" sample reconstruction of figure 4.3.63.

Parameter	*Description*	*Value*
Size	Number of vectors	200
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.63 shows the third group of results for the "pinelumb" sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first image (50 epochs) displays an almost perfect copy of the original sample, but it has few imperfections in the upper section. The second image (100 vectors) shows a decreasing in quality, some stripes are wider and the colour differs from the original texture. The third result (200 epochs) is not as accurate as the previous two results; its stripes are thicker and darker than the originals.

The last test for this sample is calculated with an even higher amount of vectors, 300. Table 4.3.33 shows the rest of parameters used for the reconstruction and figure 4.3.64 displays the obtained results.

Table 4.3.33 Parameters for the "pinelumb" sample reconstruction of figure 4.3.64.

Parameter	*Description*	*Value*
Size	Number of vectors	300
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

a) b) c) d)

Figure 4.3.64 shows the fourth group of results for the "pinelumb" sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The quality of this set of pictures is remarkable. The three images are exact copies of the original. No noticeable differences are appreciated. The stripes widths as well as their colour shades are like the original sample. It is observed that this texture is easy to reconstruct, since in three of the four groups of results, at least one image is almost an exact copy of the original texture. However, working with 300 vectors throws excellent results.

The following texture is a brick sample shown in figure 4.3.65. This texture, although regular, has the inconvenience of the joints between the bricks. Its histograms (fig.4.3.66) show the colour complexity of this image.

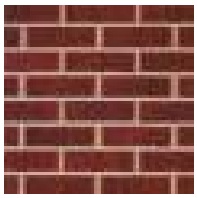
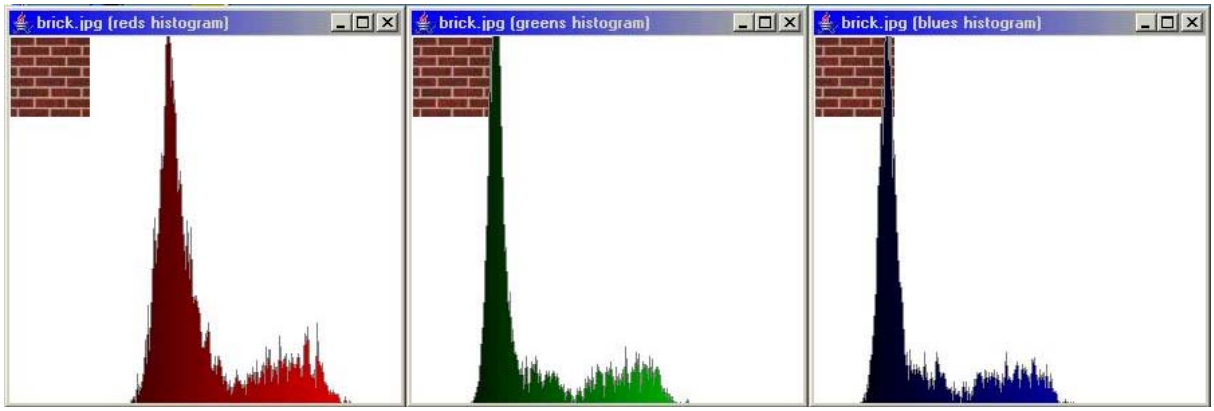


Figure 4.3.65 Brick sample.



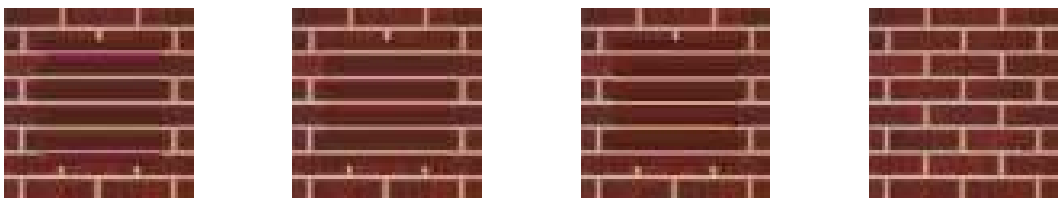
a) b) c)

Figure 4.3.66 Histograms for the bricks sample. a)red histogram, b) green histogram and c) blue histogram.

Its colour spectra are not as simple as the ones of the wooden texture; its colour ranges are also wider, but the three have well defined maximums. The first experiment runs with 50 initial vectors and a working area of 5x5 pixels. Table 4.3.34 shows the rest of parameters used for the restoration. Figure 4.3.67 shows the obtained results.

Table 4.3.34 Parameters for the brick sample reconstruction of figure 4.3.67.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

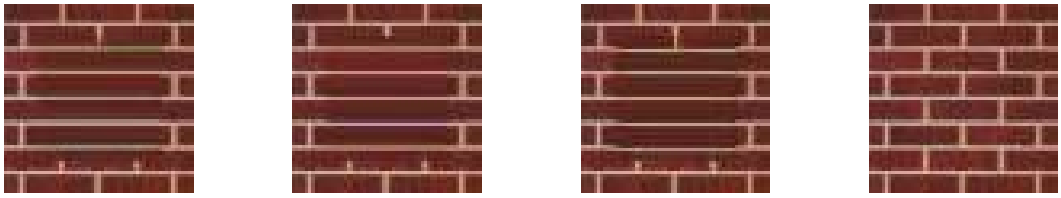
Figure 4.3.67 shows the first group of results for the brick sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

As observed, the three results are good reconstructions of the original sample. Its colour shades in bricks and joints are almost like the originals. The slight difference is that the vertical joints located in the middle section are not present anymore. Since the reconstruction pulls the image from the margin in a horizontal way, the middle section joints are not considered and only large bricks are drawn instead.

The next experiment runs with a higher number of vectors, 100 and a bigger working area, 7x7 pixels. Table 4.3.35 shows the rest of parameters. The results for this analysis are displayed in figure 4.3.68.

Table 4.3.35 Parameters for the brick sample reconstruction of figure 4.3.68.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

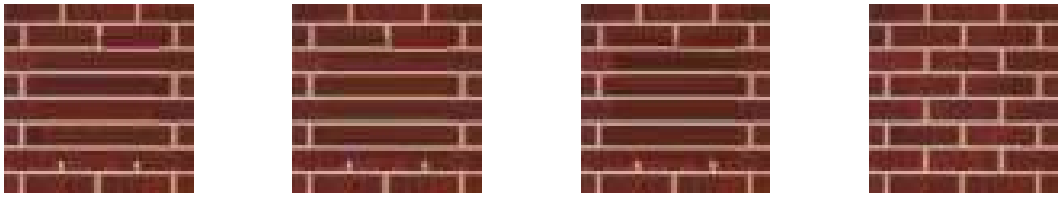
Figure 4.3.68 shows the second group of results for the brick sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The quality obtained in these results is higher than the previous obtained with 50 vectors and 5x5 pixels working area. The slight irregularities within the bricks are replaced with a solid colour that nears to the dominant colour of the original sample. The joints colour approaches even more than in the previous results (fig.4.3.67).

In the next experiment, the number of vectors is increased to 200, while the working area remains the same, 7x7 pixels. Table 4.3.36 shows the parameters and figure 4.3.69 its obtained results.

Table 4.3.36 Parameters for the brick sample reconstruction of figure 4.3.69.

Parameter	*Description*	*Value*
Size	Number of vectors	200
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

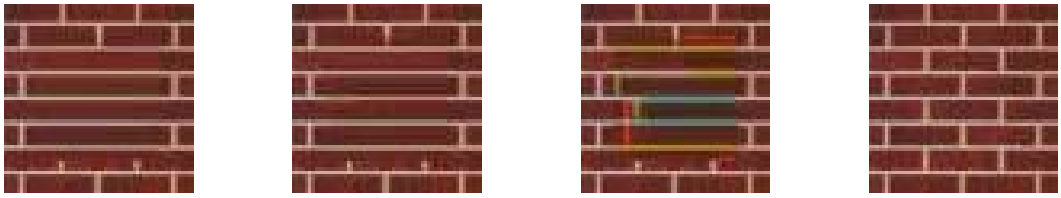
Figure 4.3.69 shows the third group of results for the brick sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first result (50 epochs) traces, in the upper and lower parts, brick patterns, with joints included. In the middle section remains the filling with large bricks. The second and third results (100, 200 epochs) show very similar images to the obtained in 50 epochs training (fig.4.3.69a). They display slight irregularities but they are not noticeable.

The last experiment with this brick sample has 300 initial vectors and its working area is reduced to 5x5 pixels. The other parameters are shown in table 4.3.37. Figure 4.3.70 displays the obtained results.

Table 4.3.37 Parameters for the brick sample reconstruction of figure 4.3.70.

Parameter	*Description*	*Value*
Size	Number of vectors	300
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

Figure 4.3.70 shows the fourth group of results for the brick sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first and second results (50 and 100 epochs) are slight better than the obtained in previous experiments (fig.4.3.67 to fig.4.3.69). The brick colour is not solid anymore, but it has also slight shade variations, like in the original.

The joints colour nears also the original sample. The third image (200 epochs) traces an intermediate joint in the middle section, but its colour shade is out of range. An accurate reconstruction of the brick sample is possible with vector quantization, but the inner part is not restored exactly as the original. The results are, however, accurate brick patterns.

4.3.2.3.4 Fourth Category: Irregular Patterned Textures

The next image to analyse is a sample of Jupiter surface (see figure 4.3.71).

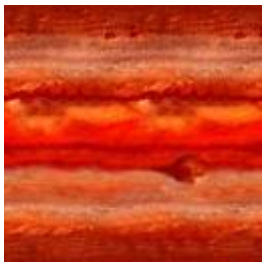


Figure 4.3.71 Jupiter sample.

The most complex image to analyse is this irregular sample. The colour shades have great contrast changes and almost no regular pattern is found. Figure 4.3.72 shows the histograms for this sample and gives an approximation of its colour complexity.

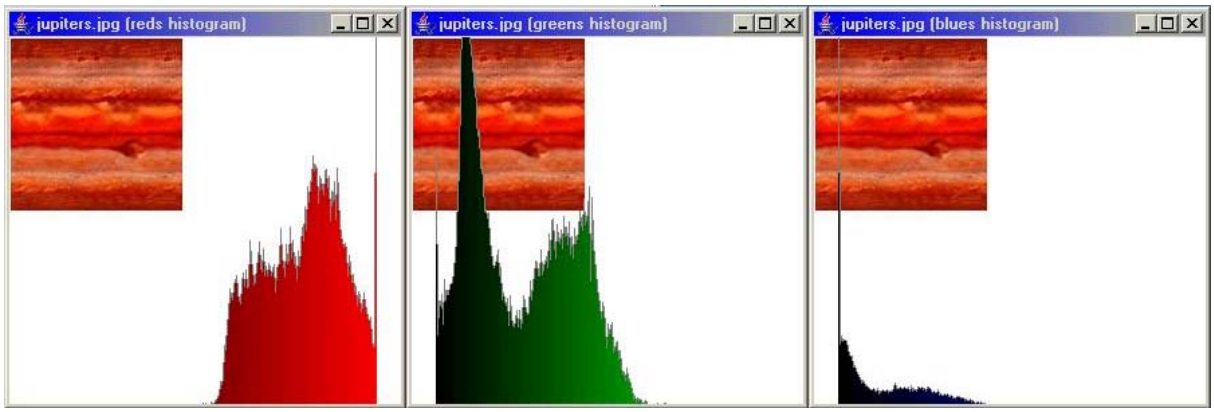


Figure 4.3.72 Histograms for Jupiter sample. a)red histogram, b) green histogram and c) blue histogram.

As observed, its colour spectra are more complex than the analysed before. The red spectrum is wide in range and has a high amount of maximums. The green histogram is even wider in range with maximums located in the dark side of the spectrum. The blue spectrum is almost inexistent. The first experiment for this sample runs with 50 vectors and the margin is set to 35 pixels, which remains constant in the following experiments. Table 4.3.38 displays the rest of parameters. The results obtained for this reconstruction are located in figure 4.3.73.

Table 4.3.38 Parameters for the brick sample reconstruction of figure 4.3.73.

Parameter	*Description*	*Value*
Size	Number of vectors	50
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200

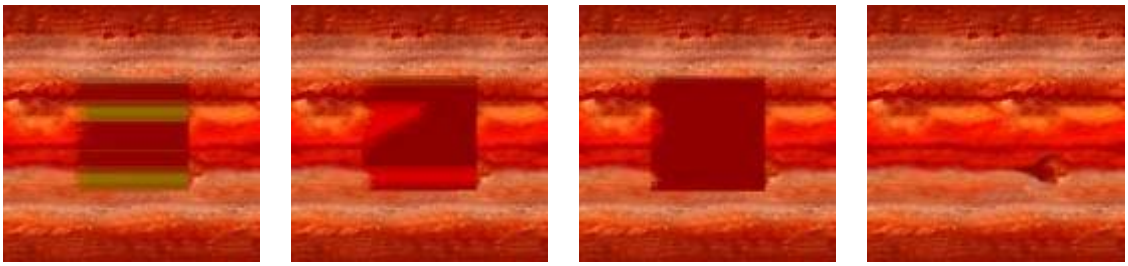


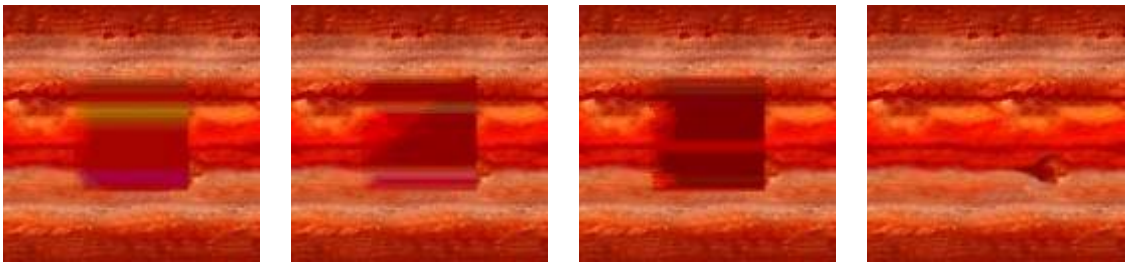
Figure 4.3.73 shows the first group of results for Jupiter sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The results are better than in the previous section (4.3.1), but still far from the original texture. The first image (50 epochs) shows a low quality reconstruction. In the second result (100 epochs), regular stripes are traced, but the colour shades differ from the original. The third image (200 epochs) fills the area with one solid colour.

The next experiment is calculated with 100 vectors, to acquire more related information. The other parameters are shown in table 4.3.39. The results obtained for these new parameters are shown in figure 4.3.74.

Table 4.3.39 Parameters for the brick sample reconstruction of figure 4.3.74.

Parameter	*Description*	*Value*
Size	Number of vectors	100
Dim	Dimension of the vectors area	3x3 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

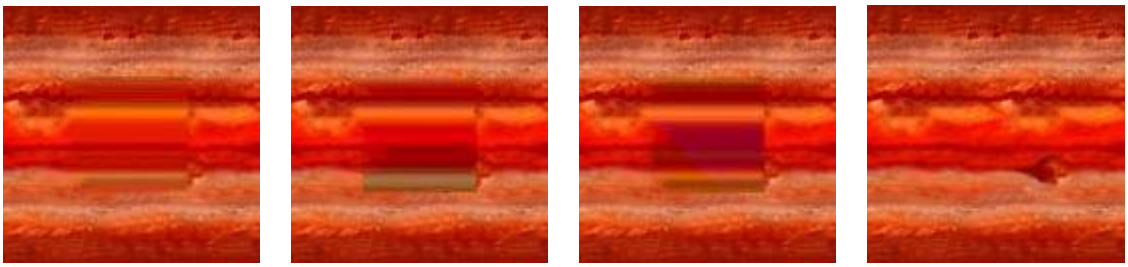
Figure 4.3.74 shows the second group of results for Jupiter sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first image (50 epochs) traces horizontal stripes, but only in the upper part. The second result (100 epochs) traces also stripes, but their colour and width do not correspond to the Jupiter surface. The third image (200 epochs) displays a different pattern from the other two results, but still far from the original sample.

The next experiment for this sample is calculated with the values shown in table 4.3.39. Vector number and working area are increased for this reconstruction. Figure 4.3.75 displays the obtained results.

Table 4.3.39 Parameters for the brick sample reconstruction of figure 4.3.75.

Parameter	*Description*	*Value*
Size	Number of vectors	200
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

Figure 4.3.75 shows the third group of results for Jupiter sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

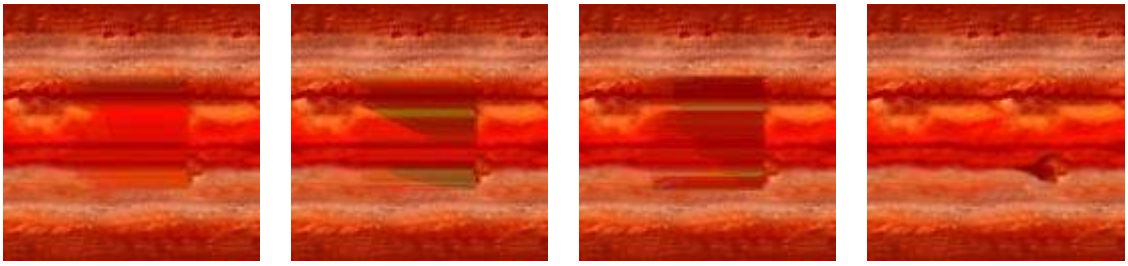
As observed, the first image (50 epochs) shows a regular pattern that is very similar to the original texture; the irregularities are not present in the reconstruction, but the colour shades and stripes width correspond to the Jupiter surface.

The second image (100 epochs) shows a lower quality restoration, the main red stripes are represented, but the bottom part of the reconstruction differs in colour from the original. The third image (200 epochs) displays some stripes similar to the original texture, but only in the upper section of area to restore.

The next experiment is calculated with 300 vectors and the working area is reduced to 5x5 pixels. Table 4.3.40 shows the rest of parameters used for this restoration. The obtained results for this analysis are displayed in figure 4.3.76.

Table 4.3.40 Parameters for the brick sample reconstruction of figure 4.3.76.

Parameter	*Description*	*Value*
Size	Number of vectors	300
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50,100 and 200
Alpha	Training factor	1/50, 1/100 and 1/200



a) b) c) d)

Figure 4.3.76 shows the fourth group of results for Jupiter sample, a) shows the result with 50 epochs, b) with 100 epochs, c) with 200 epochs and d) original sample.

The first reconstruction (50 epochs) shows a regular pattern with great similarity to the Jupiter surface. But the main characteristic of this sample is the presence of irregularities, so the image reconstruction differs from the original. This is a very accurate restoration; the differences between the restored area and the original sample are almost imperceptible. The second result (100 epochs) traces also a patterned image, whose colour shades are not distant from the original sample, but the colour shades differ in the second half of the restoration area. The last result (200 epochs) decreases in quality. The stripes are not present anymore and the colour shades are not even similar to the ones present in the original sample.

It is obvious that the results obtained in the second section are better than in the first section. The second method is the best suited to reconstruct regular images. The weighted distance is a decisive factor, but even more important factor, is the change of the working area used for the image reconstruction. This change in the reconstruction allows the image to generate whole textures, not only half, as shown in the first section. The patterns near even more the original sample thanks to the weighted distance when evaluating the vectors.

The second method is appropriate to reconstruct monochromatic, regular and pattern regular textures, but the irregular patterned images suffer changes in the restoration. However, if the area to restore is narrow, even an irregular texture can be filled with some regular pattern and the result remains satisfactory.

4.4 Comparison Between Co-occurrence Matrices and Vector Quantization

Following are some comparisons between the results obtained in both methods. Since plain and regular textures are perfectly learned and restored with both options, they are not analysed. Only regular and irregular patterned images are considered for the comparison.

The first image to compare is the pinelumb texture. Table 4.4.1 shows the parameters used in the co-occurrence matrices method; table 4.4.2 displays the parameters for the vector quantization learning process. The results and the original sample are displayed in figure 4.4.1.

Table 4.4.1 Parameters for the co-occurrence matrices restoration of the "pinelumb" sample.

Parameter	*Description*	*Value*
Margin	Margin of the original sample	10 pixels
Ver*2-1	Height of the working area (4 sectors)	3 pixels
Hor*2-1	Width of the working area (4 sectors)	3 pixels
Maxentred	Red entropy filter	0.012
Maxentgreen	Green entropy filter	0.016
Maxentblue	Blue entropy filter	0.014

Table 4.4.2 Parameters for the vector quantization restoration of the "pinelumb" sample.

Parameter	*Description*	*Value*
Margin	Margin of the original sample	10 pixels
Size	Number of vectors	300
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50
Alpha	Training factor	1/50

a) Co-occurrence matrices restoration method.

b) Original "pinelumb" sample.

c) Vector quantization restoration method.

Figure 4.4.1 Comparison between co-occurrence matrices and vector quantization reconstructions. a)Co-occurrence matrix restoration, b) original sample, c) vector quantization restoration.

As observed in previous figure, both methods build a nearly exact copy of the original sample. Vector quantization learning and reconstruction processes are faster than the ones from the co-occurrence matrices. In the first method (co-occurrence matrices), the calculation time is high due to the number of matrices that are processed. The entropy filters accelerate the learning process, because not all the matrices are considered for the restoration.

In the second method (vector quantization), although the number of vectors is high, the process does not need the same amount of time for the learning and reconstruction, because the distribution functions are created directly from the initial trained vectors. Table 4.4.3 shows the size (in Bytes) of the original sample and its reconstructions.

Table 4.4.3 Size of the "pinelumb" sample and its restorations.

Observed Image	*Bytes*
Original sample	43200
Restored area	30000
Maximum needed for the CM restoration	1454
Maximum needed for the VQ restoration	900

As the previous table shows, the co-occurrence matrix method needs more bytes to store the learned information from the "pinelumb" sample than the vector quantization method. The maximum obtained for the co-occurrence matrices is the sum of co-occurrences present in the matrices that are accepted by the entropy filter. This sum includes the three colour channels. For the vector quantization is easier to calculate. It is the number of vectors multiplied by the three colours. The learning process only consider the middle value of the vector area.

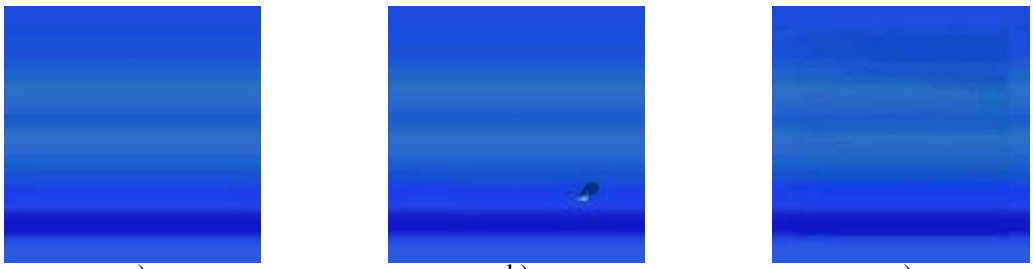
The next texture to compare is the Neptune surface sample. Tables 4.4.4 and 4.4.5 show the parameters for the co-occurrence matrices method and vector quantization restoration, respectively. Figure 4.4.2 the results obtained in both methods and the original sample.

Table 4.4.4 Parameters for the co-occurrence matrices restoration of the Neptune sample.

Parameter	*Description*	*Value*
Margin	Margin of the original sample	10 pixels
Ver*2-1	Height of the working area (4 sectors)	4 pixels
Hor*2-1	Width of the working area (4 sectors)	4 pixels
Maxentred	Red entropy filter	0.015
Maxentgreen	Green entropy filter	0.018
Maxentblue	Blue entropy filter	0.017

Table 4.4.5 Parameters for the vector quantization restoration of the Neptune sample.

Parameter	*Description*	*Value*
Margin	Margin of the original sample	10 pixels
Size	Number of vectors	300
Dim	Dimension of the vectors area	7x7 pixels
Epochs	Number of iterations	50
Alpha	Training factor	1/50



a) Co-occurrence matrices restoration method.

b) Original Neptune surface sample.

c) Vector quantization restoration method.

Figure 4.4.2 Comparison between co-occurrence matrices and vector quantization reconstructions. a)Co-occurrence matrix restoration, b) original sample, c) vector quantization restoration.

Both methods deliver very accurate restorations of the image, the slight irregularity is covered in both experiments. The first method restoration (fig.4.4.2a) is slightly better than the vector quantization reconstruction.

In the second experiment (fig.4.4.2c) in the upper part are some colour shades that differ slightly from the original sample, while the result obtained with co-occurrence matrices is a perfect replica of the sample without the irregularity. Table 4.4.6 shows the analysis of bytes needed to store the learned texture and the size of the Neptune sample as well as its reconstructions.

Table 4.4.6 Size of the Neptune sample and its restorations.

Observed Image	*Bytes*
Original sample	49152
Restored area	34992
Maximum needed for the CM restoration	5416
Maximum needed for the VQ restoration	900

As observed, the vector quantization achieves a better compression of the image features, however, the restoration obtained with the co-occurrence matrices is more accurate. The execution time is very similar for both experiments with the defined parameters of tables 4.4.3 and 4.4.4.

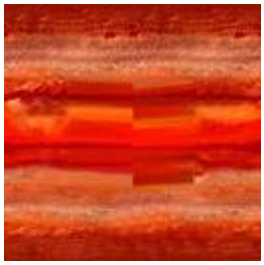
The third image to compare is the Jupiter surface sample. This is an irregular patterned image. Tables 4.4.7 and 4.4.8 display the parameters used for each experiment calculation, co-occurrence and vectors respectively. Figure 4.4.3 shows the results and the original texture.

Table 4.4.7 Parameters for the co-occurrence matrices restoration of the Jupiter sample.

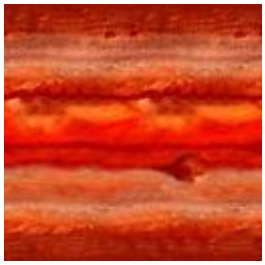
Parameter	*Description*	*Value*
Margin	Margin of the original sample	35 pixels
Ver*2-1	Height of the working area (4 sectors)	5 pixels
Hor*2-1	Width of the working area (4 sectors)	5 pixels
Maxentred	Red entropy filter	0.031
Maxentgreen	Green entropy filter	0.032
Maxentblue	Blue entropy filter	0.027

Table 4.4.8 Parameters for the vector quantization restoration of the Jupiter sample.

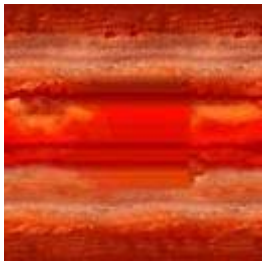
Parameter	*Description*	*Value*
Margin	Margin of the original sample	35 pixels
Size	Number of vectors	300
Dim	Dimension of the vectors area	5x5 pixels
Epochs	Number of iterations	50
Alpha	Training factor	1/50



a)
Co-occurrence matrices restoration method.



b)
Original Jupiter surface sample.



c)
Vector quantization restoration method.

Figure 4.4.3 Comparison between co-occurrence matrices and vector quantization reconstructions. a)Co-occurrence matrix restoration, b) original sample, c) vector quantization restoration.

As observed, both are good reconstructions of the original sample, but the accuracy is not as high as in the previous textures. Vector quantization method (fig.4.4.3c) creates a regular pattern that looks close enough to the original texture.

Co-occurrence matrices method displays a better reconstruction because some of the irregularities are semi-restored; the dark spot in the middle of the image is not reconstructed in its totality, but the colour shades are present in the restoration. Table 4.3.9 displays the relation between compressed texture (learned features), original sample and its restorations.

Table 4.4.6 Size of the Jupiter sample and its restorations.

Observed Image	*Bytes*
Original sample	49152
Restored	10092
Maximum needed for the CM restoration	4140
Maximum needed for the VQ restoration	900

As the previous table shows, the co-occurrence matrix needs a higher amount of learned information for an accurate restoration of the image. The execution time for this sample learning and reconstruction processes is shorter with vector quantization than the needed with the co-occurrences matrices.

As observed in the results of the three previous textures, both methods are suitable to learn different patterns and store them in the specific containers (matrices or vectors). It is also possible the restoration of complex textures, but the results are more accurate and the compression is higher with simpler textures.

5 Conclusions

As observed in previous section (4.4), both methods, co-occurrence matrices and vector quantization are suitable for the stochastic learning of textures. Plain textures and regular textures can be flawlessly restored. There are no specific parameters to choose because of the simplicity of the texture.

For regular patterned textures both methods are accurate, when choosing the correct combination of parameters for the learning and restoration processes. For complex textures, the size of working area must be large, in order to collect enough information for the restoring. An augment in the working area causes also an increment on the execution time; however the results obtained are very accurate. Vector quantization method is faster than co-occurrence matrix.

The learning of irregular patterned textures is also realizable, but its restoration is only possible in narrow areas, because both methods reconstruct the specified area with a regular pattern similar to the neighbourhood. In conclusion, co-occurrence matrices and vector quantization are reliable and accurate options for the learning and reconstruction of textures.

Possible applications of texture learning are:

- Reconstruction and restoration of textures.
- Comparison of textures.
- Creation of similar textures, variation of textures, for example in Animation.

Further necessary research activities:

- Minimisation of storage costs for learned textures.
- Analysis and comparison of further learning procedures based in Neural Networks, similar to Vector Quantization, like Kohonen Self Organized Maps or Neural Gas.
- Development of a model for texture memory.

6 List of Algorithms

7 List of References

[1]. D. M. SZAFLARSKI. *How we see: The First Steps of Human Vision.* [Online]
<http://www.accessexcellence.org/AE/AEC/CC/vision_background.html>
[Last visited: November 2004]

[2]. L. R. PRADO LEON. *Psicología de la percepción.* [Online]
<http://www.cuaad.udg.mx/~lprado/portal-psic/Sistema%20visual/Anatomia%20ojo.htm>
[Last visited: November 2004]

[3]. ANATOMY OF THE EYE [Online]
<http://www.tlcvision.com/what_is_laser_vision_correction/anatomy_of_the_eye/>
[Last visited: November 2004]

[4]. G. M. MURCH. *Physiological Principles for the Effective Use of Colour*. IEEE Computer Graphics and Applications 4, (Nov. 1984), pp. 49-54.

[5]. J. C. RUSS. *The Image Processing Handbook. 3rd edition.* CRC Press LLC, 1999, pp. 1-7.

[6]. M. ORZESSEK, P. SOMMER. *ATM & MPEG-2 Integrating digital video into broadband networks*. Prentice Hall, 1998, pp 17,18.

[7]. C. TORRAS. *Computer Vision. Theory and Industrial Applications*. Springer-Verlag, 1992, pp. 1,2.

[8]. D. HEARN, M. P. BAKER, *Computer graphics. Second Edición*. Prentice Hall, 1994, pp 36-53.

[9]. WHAT IS.COM. [Online]
<http://whatis.techtarget.com/>
[Last visited: November 2004]

[10]. T. HARRIS. *How plasma displays works.* [Online]
<http://electronics.howstuffworks.com/plasma-display.htm>
[Last visited: November 2004]

[11]. J. TYSON. *How LCDs work.* [Online]
<http://electronics.howstuffworks.com/lcd.htm>
[Last visited: November 2004]

[12]. LIQUID CRISTAL DISPLAYS [Online]
<http://plc.cwru.edu/tutorial/enhanced/files/lcdindx.htm>
[Last visited: February 2004]

[13]. THIN FILM TRANSISTOR TECHNOLOGIES [Online]
<http://www.eecs.berkeley.edu/~tking/tft.html>
[Last visited: November 2004]

[14]. LZW DATA COMPRESSION [Online]
<http://www.dogma.net/markn/articles/lzw/lzw.htm>
[Last visited: November 2004]

[15]. UNISYS WEB SITE [Online]
<http://www.unisys.com/about__unisys/lzw/>
[Last visited: November 2004]

[16]. WEB GRAPHICS [Online]
<http://archive.devx.com/projectcool/developer/gzone/>
[Last visited: November 2004]

[17]. GRAPHICS FORMATS FREQUENTLY ASKED QUESTIONS [Online]
<http://www.cywarp.com/FAQ_TimeCapsules.htm>
[Last visited: November 2004]

[18]. PNG HOME PAGE [Online]
<http://www.libpng.org/pub/png/>
[Last visited: November 2004]

[19]. ISO – INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [Online]
<http://www.iso.ch/>
[Last visited: November 2004]

[20]. JPEG IMAGE COMPRESSION FREQUENTLY ASKED QUESTIONS [Online]
<http://www.faqs.org/faqs/jpeg-faq/part1/>
[Last visited: November 2004]

[21]. COLOURCUBE [Online]
<http://www.colourcube.com/index.htm>
[Last visited: February 2004]

[22]. COLOURES MUNSELL [Online]
<http://www.mindspring.com/~larinc/sbr/replications/historical/munsell.htm>
[Last visited: February 2004]

[23]. MUNSELL COLOUR. *The Universal Language.* [Online]
<http://www.munsell.com>
[Last visited: November 2004]

[24]. D. HEARN, M. P. BAKER, *Computer graphics. Second Edición*. Prentice Hall, 1994, pp 599-601.

[25]. WEB SITE TXIPINET. *Diseño de GUIs*. [Online]
<http://www.txipinet.com/gui2.php>
[Last visited: November 2004]

[26]. D. HEARN, M. P. BAKER, *Computer graphics. Second Edición*. Prentice Hall, 1994, pp 602-604.

[27]. WEB SITE ACM CROSSROADS. *Técnicas y Herramientas para Usar Color en el Diseño de la Interfaz de una Computadora* [Online]
< http://www.acm.org/crossroads/espanol/xrds3-3/colour.html>
[Last visited: February 2004]

[28]. D. HEARN, M. P. BAKER, *Computer graphics. Second Edición*. Prentice Hall, 1994, pp 605-606.

[29]. D. HEARN, M. P. BAKER, *Computer graphics. Second Edición*. Prentice Hall, 1994, pp 604-605.

[30]. D. HEARN, M. P. BAKER, *Computer graphics. Second Edición*. Prentice Hall, 1994, pp 606-608.

[31]. D. HEARN, M. P. BAKER, *Computer graphics. Second Edición*. Prentice Hall, 1994, pp 610-611.

[32]. O. KAO. *Dynamisches Retrieval von Multimedialen Daten auf parallelen Architekturen.* Habilitationsschrift TU Clausthal, 2001, pp. 28,29.

[33]. B. JAEHNE, *Digitale Bildverarbeitung*. Springer-Verlag 1997, pp. 407-410.

[34]. TEXTURE ANALYSIS. *Analyzing and Enhancing Images (Image Processing Toolbox User's Guide).* [Online]
<http://www.mathworks.com/access/helpdesk/help/toolbox/images/enhanc14.html>
[Last visited: November 2004]

[35]. J. C. RUSS. *The Image Processing Handbook. 3rd edition.* CRC Press LLC, 1999, pp. 278-282.

[36]. P. HABERAECKER, *Digitale Bildverarbeitung. Grundlagen und Anwendungen. 2. Auflage.* Carl Hansen Verlag, 1987, pp 25-30.

[37]. B. JAEHNE, *Digitale Bildverarbeitung*. Springer-Verlag 1997, pp. 82.

[38]. J. C. RUSS. *The Image Processing Handbook. 3rd edition.* CRC Press LLC, 1999, pp. 233-234.

[39]. O. KAO. *Dynamisches Retrieval von Multimedialen Daten auf parallelen Architekturen.* Habilitationsschrift TU Clausthal, 2001, pp. 138.

[40]. P. HABERAECKER, *Digitale Bildverarbeitung. Grundlagen und Anwendungen. 2. Auflage.* Carl Hansen Verlag, 1987, pp 30-31.

[41]. GREY LEVEL CO-OCCURRENCE MATRIX (GLCM). *Analyzing and Enhancing Images (Image Processing Toolbox User's Guide).* [Online]
< http://www.mathworks.com/access/helpdesk/help/toolbox/images/enhanc16.html#29651>
[Last visited: November 2004]

[42]. P. HABERAECKER, *Digitale Bildverarbeitung. Grundlagen und Anwendungen. 2. Auflage.* Carl Hansen Verlag, 1987, pp 31-33.

[43]. O. KAO. *Dynamisches Retrieval von Multimedialen Daten auf parallelen Architekturen.* Habilitationsschrift TU Clausthal, 2001, pp. 155-157.

[44]. PLANETARY TEXTURES FOR MODEL CONSTRUCTION. [Online]
<http://inkido.indiana.edu/a100/planetary_textures.html>
[Last visited: November 2004]

[45]. COMP.COMPRESSION FREQUENTLY ASKED QUESTIONS (part 2/3) [Online].
<http://www.faqs.org/faqs/compression-faq/part2/>
[Last visited: November 2004]

[46]. A.ZELL, *Simulation Neuronaler Netze.* Addison-Wesley, 1994, pp 171-178.

[47]. VECTOR QUANTIZATION. [Online]
<http://www.data-compression.com/vq.html>
[Last visited: November 2004]

[48]. ARTIFICIAL NEURAL NETWORKS. *IV Vector Quantization.* [Online]
<http://www.dice.ucl.ac.be/~verleyse/lectures/elec2870/elec2870.htm>
[Last visited: November 2004]

[49]. VECTOR QUANTIZATION. [Online]
<http://www.geocities.com/mohamedqasem/vectorquantization/vq.html>
[Last visited: November 2004]

[50]. LEARNING VECTOR QUANTIZATION VISUALIZATION. [Online]
< http://fuzzy.cs.uni-magdeburg.de/~borgelt/doc/lvqd/>
[Last visited: November 2004]